Uklanjanje impulsnog šuma iz slike primjenom selektivnog medijan filtra s detekcijom šuma

Matijas, Adriana

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

UKLANJANJE IMPULSNOG ŠUMA IZ SLIKE PRIMJENOM SELEKTIVNOG MEDIJAN FILTRA S DETEKCIJOM ŠUMA

Diplomski rad

Adriana Matijas

Osijek, 2016.

SADRŽAJ

1. UVOD

Razvoj multimedijske tehnologije je posljednjih godina u vrlo velikom porastu. Jedan od najčešćih medija koji se pojavljuju u multimedijskim sustavima su digitalne slike. Međutim digitalne slike su često degradirane šumom koji se javlja prilikom njihovog nastanka i prijenosa. Usporedo sa sve većom upotrebom digitalnih slika razvijeni su i različiti algoritmi koji se koriste za uklanjanje šuma iz njih [1]. Glavni cilj upotrebe tih algoritama je restaurirati sliku tako da prilikom uklanjanja određenog šuma ona ne bude bitno izmjenjenja i da njezini detalji ostanu sačuvani u što većoj mjeri. U ovom radu predstavit će se pojam digitalna slika, te česti modeli šuma, s naglaskom na dva najčešća modela impulsnog šuma, a to su sol i papar šum te šum nasumičnih vrijednosti, poznatiji pod engleskim terminom *random-valued*. Predstavit će se neke od metoda za njihovo uklanjanje, dok će se najviše pažnje posvetiti selektivnom adaptivnom medijan filtru s detekcijom šuma i usporedbi njegove efikasnosti s efikasnošću običnog medijan filtra. U radu je predstavljen problem koji se pojavljuje prilikom uklanjanja šuma iz slike primjenom medijan filtara, a to je da se kod konvencionalnih pristupa medijan filtri primjenjuju na sve elemente slike, bez obzira na to jesu li oni oštećeni ili ne što rezultira neželjenim i naravno nepotrebnim zamućenjem slike. Taj problem se može riješiti korištenjem preciznih algoritama za detekciju šuma koji će zajedno s adaptivnim medijan filtrom uspješno detektirati sve oštećene elemente slike te ih u konačnici i ukloniti. U ovom radu u programskom paketu MATLAB implementiran je selektivni medijan filtar s detekcijom šuma. Ako je kod uspješno implementiran u MATLAB-u u drugim programskih jezicima ga isto tako nije teško implementirati, ali se mora paziti koje se ugrađene funkcije MATLAB-a koriste. Samo uklanjanje šuma iz slike pomoću MATLAB-a bilo je uspješno, ali vrlo sporo, posebno kod jako degradiranih slika. Za primjenu u realnim sustavima osim samog uspjeha prilikom uklanjanja šuma potrebno je i osigurati dovoljno brzu obradu slike odnosno uklanjanje šuma. Zbog toga je bilo vrlo bitno izabrati programski jezik u kojemu će šum biti uklonjen što većom mogućom brzinom. Zbog karakteristika C++ programskog jezika u ovom radu adaptivni medijan filtar s detekcijom šuma nakon implementacije u MATLAB-u implementiran je upravo u njemu. Sam C++ programski jezik nije dovoljan za kreiranje ovakve vrste koda, pa je korištena OpenCV biblioteka koja je kreirana posebno za ovakve vrste primjene (računalni vid, obrada slike). Kako bi se napravljeni kod mogao nositi i s postotkom šuma od čak 90% u ovom radu je analiziran BDND algoritam, te je primijenjen na degradiranu sliku.

U drugom poglavlju opisan je pojam digitalna slika i njezina podjela na binarne, monokromatske i višekanalne slike. Predstavljeni su i modeli šuma koji se pojavljuju u digitalnim slikama: Gaussov, Rayleighov, gamma, eksponencijalni, uniformni, periodični te impulsni šum.

Također dane su formule za funkcije gustoće vjerojatnosti amplitude za svaki navedeni šum. Prikazani su i primjeri na kojima je test slika oštećena određenim šumom.

U trećem poglavlju opisano je uklanjanje impulsnog šuma, te kratak uvod u razvoj raznih inačica medijan filtra (MF). Detaljno je opisan standardni medijan filtar i njegov algoritam. Opisan je i selektivni adaptivni medijan filtar (AMF) te algoritam po kojemu je napravljen program za implementaciju ovog filtra. Nakon izrade AMF-a u MATLAB programskom paketu AMF je izveden i kao C++ program. Detaljno je opisan način konfiguracije operacijskog sustva i povezivanja projekta s bibliotekama. Prikazan je rezultat testiranja gdje je filtrirana Lena slika oštećena s 31% šuma. U ovom poglavlju obrađuje se i BDND algoritam. Izveden je samo u MATLAB programskom paketu zbog težine implementacije BDND funkcije u C++. Efikasnost uklanjanja impulsnog šuma tipa sol i papar primjenom MF, AMF i BDND pokazana je na primjeru slike s 30, 50, 70 i 90% oštećenja. Napravljena je i usporedba efikasnosti MF, AMF i BDND pomoću PSNR vrijednosti. Izveden je i iterativni postupak uklanjanja šuma. Uspoređene su i složenosti MF, AMF i BDND algoritma za slike Lena, Baboon, Pears i Barbara i to uz oštećenje od 20, 50 i 70%.

U posljednjem poglavlju dan je zaključak koji je izveden na temelju svih predstavljenih rezultata.

2. ŠUM U DIGITALNIM SLIKAMA

Glavni izvori šuma u digitalnim slikama se pojavljuju prilikom njihovog nastanka i/ili prijenosa. Performanse slikovnih senzora su često pod utjecajem raznih faktora, od uvjeta u okolini prilikom nastanka slike pa sve do kvalitete samih elemenata senzora. Na primjer kod slika nastalih CCD kamerom glavni faktori nastanka šuma su razine svjetlosti i temperature senzora. S druge strane, prilikom prijenosa, slike su oštećene prvenstveno zbog smetnji u kanalu koji se koristi za prijenos. Na primjer slika koja se prenosi putem bežične mreže može biti oštećena zbog posljedice udara groma ili neke druge atmosferske smetnje, [2].

2.1. Digitalna slika

Slika je dvodimenzionalna funkcija *f(x,y),* gdje njezine vrijednosti x i y mogu poprimiti bilo koje vrijednosti od 0 (crna) do 255 (bijela). Kako bi se slika uopće mogla kompjuterski obrađivati potrebno ju je imati u jednom od digitalnih formata.

Digitalna slika predstavlja funkciju *f(x,y)* i sastoji se od elemenata slike čija je pozicija definirana sa x i y. Svaki se takav element čija je pozicija u slici definirana sa x i y, naziva element slike. S obzirom na to kakve sve vrijednosti elementi slike mogu poprimiti digitalne slike se dijele na:

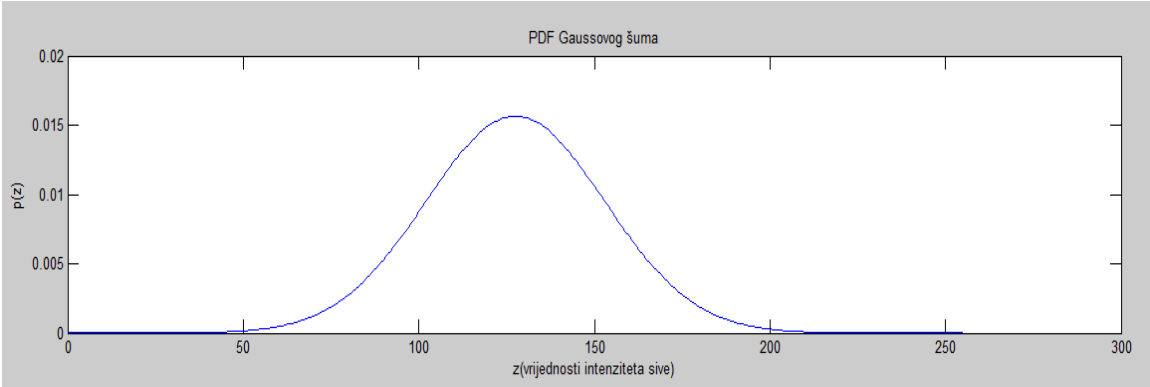
1) Binarne slike kod kojih svaki element slike može poprimiti ili 1 ili 0.
2) Monokromatske slike sastoje se od elemenata slike koji poprimaju vrijednosti samo jedne boje (najčešće sive). Kod njih je broj nijansi određen dubinom boje koji se označava s potencijama broja dva. Tako 4-bitne dubine boja daju 16 nijansi, 8-bitne dubine boja daju 256 nijansi, itd.
3) Višekanalne slike (slike u boji) za svaki element slike koriste više od jedne vrijednosti. Najčešće se koristi za svaki kanal slike 8-bitna dubina boje za svaki kanal (vrijednosti od 0 do 255) što u konačnici daje oko 16,77 milijuna boja, [3].

2.2. Gaussov šum

Funkcija gustoće vjerojatnosti amplitude Gaussovog šuma dana je izrazom:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(z-\bar{z})^2}{2\sigma^2}} \tag{2-1}$$

Prikaz ove funkcije vidljiv je na slici 2.1.



Sl. 2.1. Funkcija gustoće vjerojatnosti amplitude Gaussovog šuma

Na slici 2.2. prikazane su (s lijeva na desno) originalna slika i slika degradirana Gaussovim šumom.

a) Izvorna slika b) Degradirana slika

Sl. 2.2.Izvorna slika i slika degradirana s 60% gaussovog šuma

2.3. Rayleighov šum

Funkcija gustoće vjerojatnosti amplitude Rayleighovog šuma dana je slijedećim izrazom:

$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & \text{for z } \geq \text{a} \\ 0 \text{ for z} < a \end{cases} \quad (2\text{-}2)$$

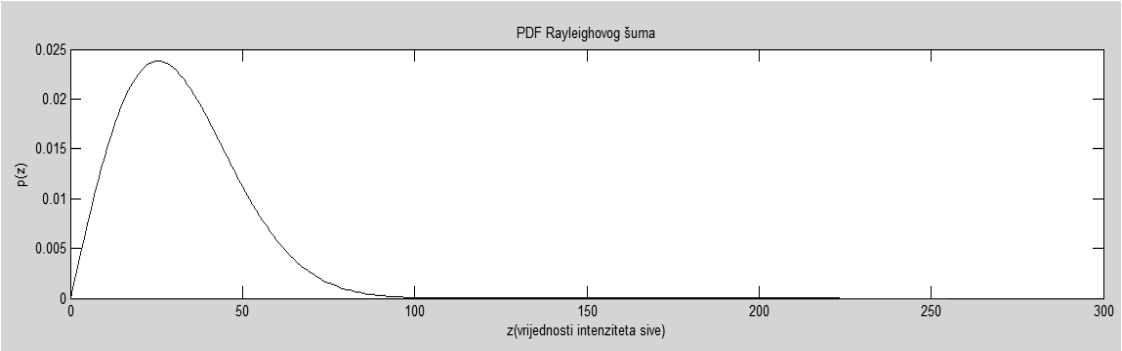
Srednja vrijednost i varijanca dane su izrazima:

$$\overline{z} = \text{a} + \sqrt{\pi b/4} \tag{2-3}$$

i

$$\sigma^2 = \frac{b(4-\pi)}{4} \tag{2-4}$$

Prikaz ove funkcije vidljiv je na slici 2.3.



Sl. 2.3. Funkcija gustoće vjerojatnosti amplitude Rayleighovog šuma

Na slici 2.4. prikazane su (s lijeva na desno) originalna slika i slika degradirana Rayleighovim šumom.

a) Izvorna slika b) Degradirana slika

Sl. 2.4.Izvorna slika i slika s 60% Rayleighovog šuma

2.4. Erlang (Gamma) šum

Funkcija gustoće vjerojatnosti amplitude Erlangovog šuma dana je slijedećim izrazom:

$$p(z) = \begin{cases} e^{-az}\frac{a^b z^{b-1}}{(b-1)!} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases} \tag{2-5}$$

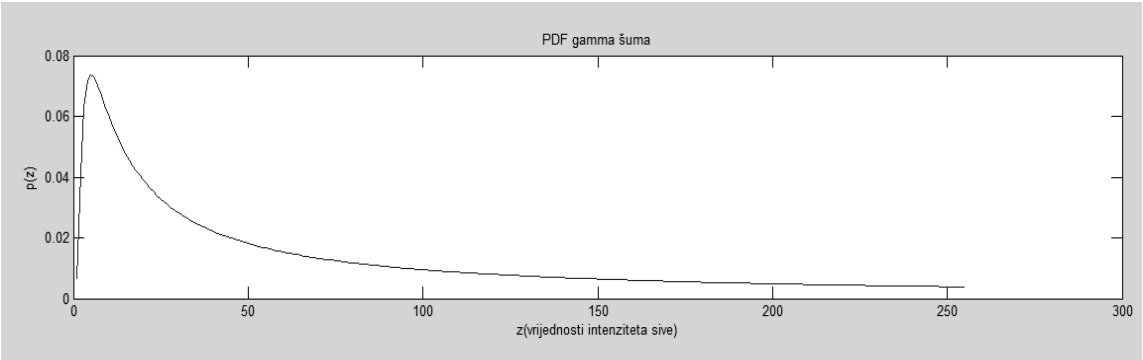
Ako je a > 0, b je pozitivan cijeli broj. Srednja vrijednost i varijanca su dane izrazom:

$$\overline{z} = \frac{b}{a} \tag{2-6}$$

i

$$\sigma^2 = \frac{b}{a^2} \tag{2-7}$$

Prikaz ove funkcije vidljiv je na slici 2.5.



Sl. 2.5. Funkcija gustoće vjerojatnosti amplitude Erlangovog šuma

Na slici 2.6. prikazane su (s lijeva na desno) originalna slika i slika degradirana Erlangovim šumom.

a) Izvorna slika b) Degradirana slika

Sl. 2.6. Izvorna slika i slika s 60% Erlangovog šuma

2.5. Eksponencijalni šum

Funkcija gustoće vjerojatnosti amplitude eksponencijalnog šuma dana je izrazom:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(z-\overline{z})^2}{2\sigma^2}} \tag{2-8}$$

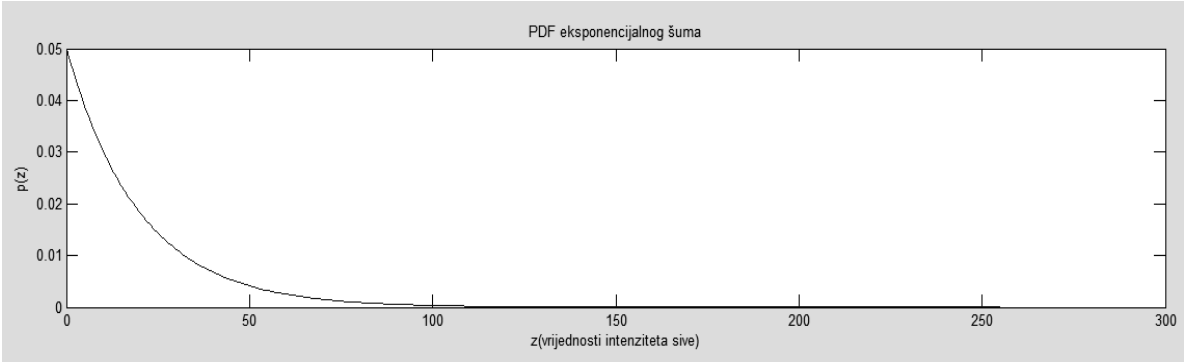
Gdje je a > 0. Srednja vrijednost i varijanca dane su izrazima:

$$\overline{z} = \frac{1}{a} \tag{2-9}$$

i

$$\sigma^2 = \frac{1}{a^2} \tag{2-10}$$

Prikaz ove funkcije vidljiv je na slici 2.7.



Sl. 2.7. Funkcija gustoće vjerojatnosti eksponencijalnog šuma

Na slici 2.8. prikazane su (s lijeva na desno) originalna slika i slika degradirana eksponencijalnim šumom.

a) Izvorna slika b) Degradirana slika

Sl. 2.8. Izvorna slika i slika degradirana s 60% eksponencijalnog šuma

2.6. Uniformni šum

Funkcija gustoće vjerojatnosti amplitude uniformnog šuma dana je izrazom:

$$\begin{matrix} \frac{1}{b-a} \text{ if a } \leq \text{z } \leq \text{b} \\ 0 \; inače \end{matrix} \tag{2-11}$$

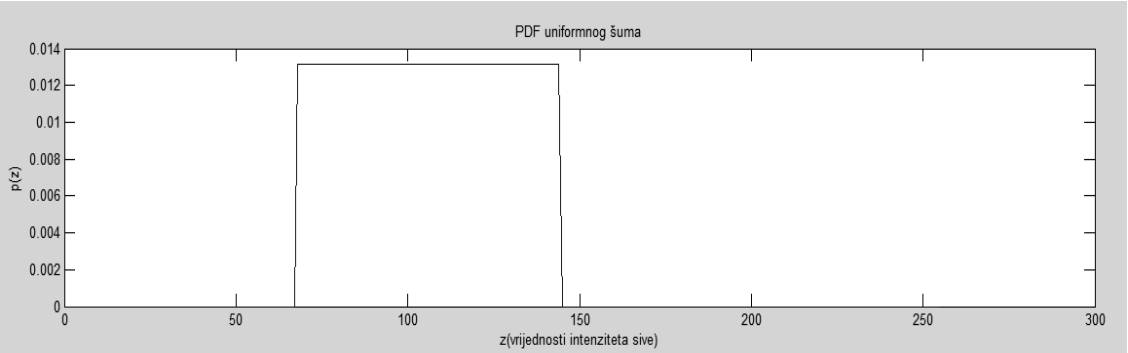
Srednja vrijednost i varijanca dane su izrazom:

$$\overline{z} = \frac{a+b}{2} \tag{2-12}$$

i

$$\sigma^2 = \frac{(b-a)^2}{12} \tag{2-13}$$

Prikaz ove funkcije vidljiv je na slici 2.9.



Sl. 2.9. Funkcija gustoće vjerojatnosti amplitude uniformnog šuma

Na slici 2.10. prikazane su (s lijeva na desno) originalna slika i slika degradirana uniformnim šumom.

a) Izvorna slika b) Degradirana slika

Sl. 2.10.Izvorna slika i slika degradirana s 60% uniformnog šuma

2.7. Periodični šum

Periodični šum se u slikama uglavnom pojavljuje zbog električnih ili elektromehaničkih smetnji prilikom nastajanja slike. Perodični šumovi se mogu značajno smanjiti filtriranjem u frekvencijskoj domeni.

2.8. Modeli impulsnog šuma

Kao što je već rečeno digitalne slike mogu biti degradirane impulsnim šumom prilikom njihovog nastanka ili prijenosa. Intenzitet impulsnog šuma može biti relativno nizak ili relativno visok. Prema tome ovakva vrsta šuma može ozbiljno promijeniti izgled slike, i uzrokovati gubitak bitnih detalja slike. To se događa jer je impuslni šum, koji se pojavljuje na skupu nasumičnih elemenata slike, uglavnom u velikom kontrastu sa svojom okolinom. Najčešće je uzrokovan nepravilnim elementima slike u senzorima kamere, neispravnim memorijskih lokacijama u sklopovlju ili prijenosom kroz kanal sa smetnjama

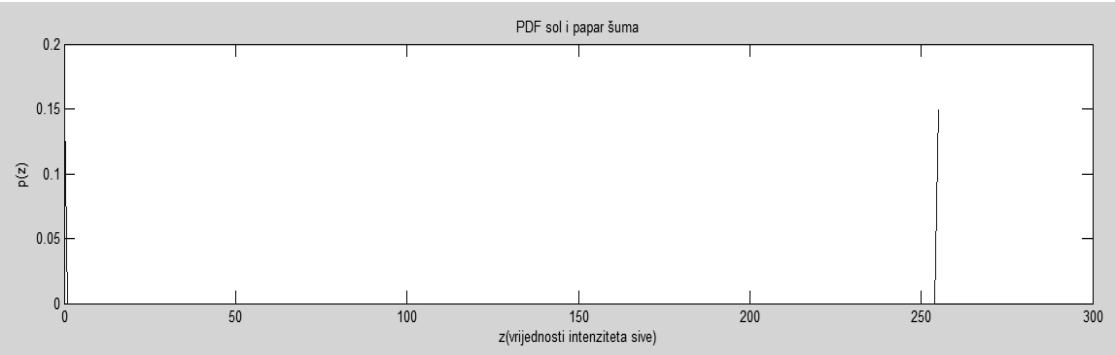
S obzirom na vrijednosti šum se može podijeliti na onaj koji je jednostavniji za uklanjanje, takozvani sol i papar (salt-and-pepper) šum te na puno složeniji za uklanjanje, šum nasumičnih vrijednosti (random-valued šum),[4].

2.8.1. Sol i papar šum

Funkcija gustoće vjerojatnosti amplitude bipolarnog impulsnog šuma dana je izrazom:

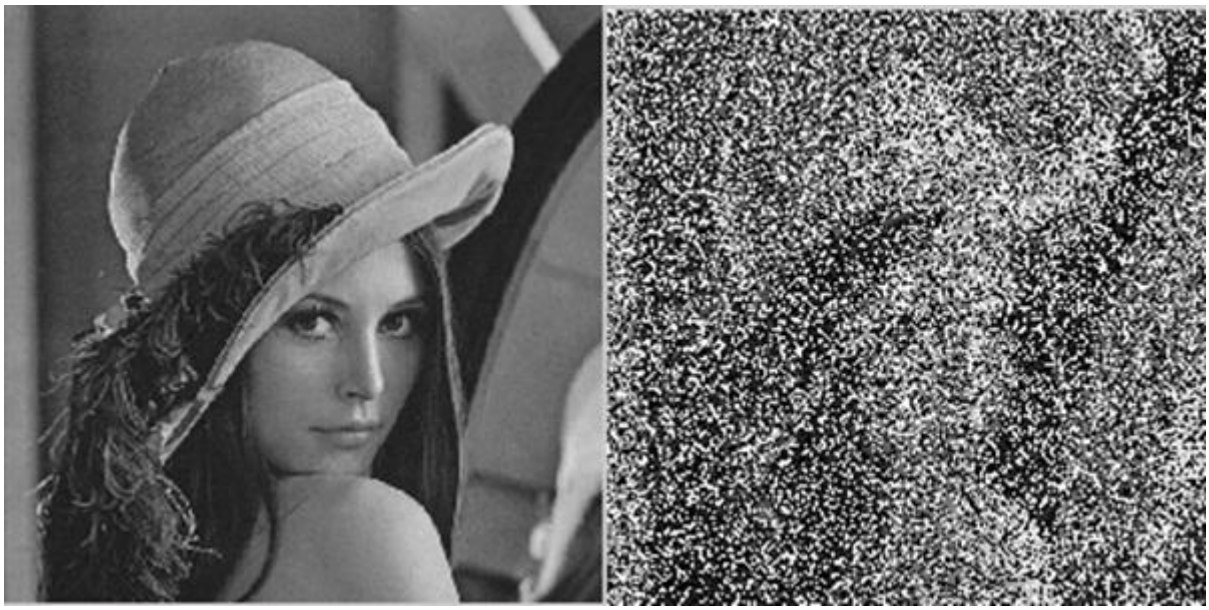
$$p(z) = \begin{cases} P_a \text{ for } z = a \\ P_b \text{ for } z = b \\ 0 \text{ inače} \end{cases} \tag{2-14}$$

Prikaz ove funkcije vidljiv je na slici 2.11.



Sl. 2.11. Funkcija gustoće vjerojatnosti amplitude sol i papar šuma

Na slici 2.12. prikazane su (s lijeva na desno) originalna slika i slika degradirana impulsnim sol i papar šumom.



a) Izvorna slika b) Degradirana slika

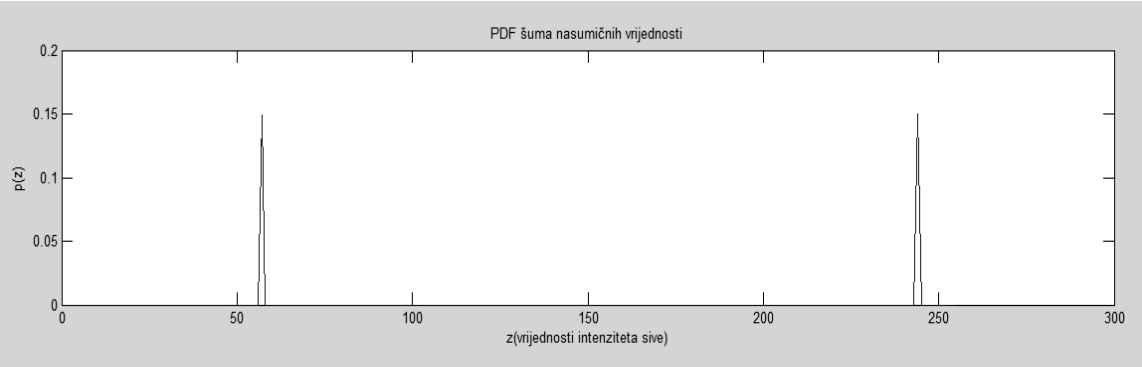
Sl. 2.12.Izvorna slika i slika degradirana s 60% sol i papar šuma

Ako je $b > a$, b će se pokazati kao svijetla točka na slici. Suprotno tome, a će se pokazati kao tamna točka. Ako je ili P_a ili P_b jednak nuli, šum će biti unipolarni. U slučaju da niti jedna od vjerojatnosti nije jednaka nuli, i posebno u slučaju ako su vjerojatnosti skoro jednake, impulsni šum će se prikazati kao čestice soli i papra koje su nasumično raspoređene po slici. Upravo zbog takvog prikaza ovaj bipolarni šum se često zove *sol i papar* šum.

Impulsni šum može biti pozitivan ili negativan. Skaliranje je obično dio procesa digitalizacije slike. Zato što je impulsno oštećenje uglavnom puno veće u usporedbi sa signalom slike, impulsni šum biva digitaliziran u vidu ekstremnih (bijelih i crnih) vrijednosti u slici. Tako je uglavnom pretpostavka da su a i b „zasićene“ vrijednosti, u smislu da su jednake minimumu i maksimumu dozvoljenih vrijednosti u digitaliziranoj slici. Kao rezultat toga, negativni impulsi se prikazuje kao crne (papar) točkice na slici, dok se pozitivni impulsi javljaju kao bijele (sol) točkice na slici. Za 8-bitnu sliku to znači da je $a = 0$ (crno) i $b = 255$ (bijela), [2].

2.8.2. Šum nasumičnih vrijednosti

Za razliku od sol i papar šuma gdje su *a* i *b* zasićene vrijednosti, koje odgovaraju minimumu i maksimumu dozvoljenih vrijenosti u digitaliziranoj slici, kod šuma nasumičnih vrijednosti *a* i *b* mogu poprimiti bilo koju vrijednost u rasponu od minimalne do maksimalne vrijednosti. Zbog toga je šum nasumičnih vrijednosti puno teže i složenije ukloniti za razliku od jednostavnog sol i papar šuma gdje su razlike u nijansama sive između elemenata slike koji su oštećeni šumom i susjednim neoštećenim elementima slike najčešće vrlo značajne.



Sl. 2.13. Funkcija gustoće vjerojatnosti amplitude *random-valued* šuma

Na slici 2.14. prikazane su (s lijeva na desno) originalna slika i slika degradirana *random-valued* šumom.

a) Izvorna slika b) Degradirana slika

Sl. 2.14.Izvorna slika i slika degradirana s 60% *random-valued* šuma

3. UKLANJANJE IMPULSNOG ŠUMA

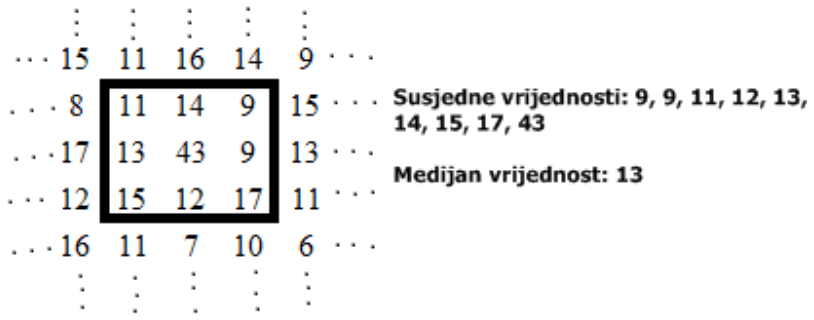
Kako bi se riješio problem pojave impulsnog šuma, ali i ostalih vrsta šuma čime bi se uvelike poboljšala kvaliteta slike, velik broj znanstvenika je radio na tom polju istraživanja, [5] Među prvim rješenjima za uklanjanje impulsnog šuma predstavljeni su linearni filtri. Međutim kako je poznato da linearni filtri mogu uzrokovati ozbiljna zamućenja slike, počela se proučavati uporaba nelinearnih filtara za uklanjanje šuma. Kao rezultat toga nelinearni filtri sada imaju široku uporabu kod filtriranja slika s obzirom na njihove mogućnosti filtriranja, u smislu prigušenja šuma te očuvanja detalja i rubova. Na temelju raznih istraživanja sada je poznato da je medijan filtar vrlo korisna tehnika kojom se uspijeva oduprijeti efektima utjecaja impulsnog šuma na sliku. Međutim, postoje razne varijacije medijan filtra. Uz standardni medijan filtar tako postoji iterativni medijan filtar, rekurzivni medijan filtar, težinski medijan filtar, usmjereni medijan filtar, filtar s prebacivanjem, selektivni adaptivni medijan filtar, a povećano zanimanje za ovo polje istraživanja pretpostavka je da se svakodnevno razvijaju nove poboljšane varijacije medijan filtra.

3.1. Standardni medijan filtar

Medijan filtar (MF) spada u skupinu nelinearnih filtara, što znači da za dvije slike A(x) i B(x) vrijedi:

$$median\,[A(x)\ +\ B(x)]\ \neq\ median\,[A(x)]\ +\ median\,[B(x)] \tag{3-1}$$

Medijan je u ovom slučaju srednja vrijednost svih vrijednosti elemenata slike u susjednom okruženju. Mora se uzeti u obzir kako to nije isto što i prosječna vrijednost. Ovaj filtar uzima u obzir svaki element slike i promatra njegove susjedne elemente slike. Umjesto da element slike zamijeni prosječnom vrijednosti susjednih elemenata slike zamijeniti će ga s medijanom tih vrijednosti. Medijan se računa tako da se prvo sortiraju vrijednosti elemenata slike numeričkim redoslijedom, te se onda promatrani element slike zamijeni elementom srednje vrijednosti. [6] Na slici 3.1. prikazan je postupak određivanja medijan vrijednosti.



Sl. 3.1.Postupak određivanja medijan vrijednosti

U slučaju da postoji paran broj vrijednosti, uzeti će se u obzir prosjek dvije srednje vrijednosti. Koraci algoritma medijan filtra su sljedeći:

1) Postaviti prozor veličine 3 × 3 oko promatranog elementa slike
2) Sortirati vrijednosti elemenata slike u susjedstvu
3) Odabrati element slike srednje vrijednosti
4) Promatrani element slike zamijeniti odabranim elementom slike srednje vrijednosti

Sam kod za standardni medijan filtar u MATLAB-u je vrlo jednostavan, jer postoji gotova naredba koja izvršava cijelo medijan filtriranje. Linija koda sa spomenutom naredbom je: B = medfilt2 (A, [m n]), gdje se izvodi medijan filtriranje matrice A, medijan filtrom veličine mxn, gdje svaki izlazni element slike sadrži srednju (medijan) vrijednost oko odgovarajućeg elementa slike u ulaznoj slici.[7]

Iako standardni medijan filtar može značajno smanjiti razinu oštećenja uzrokovanog impulsnim šumom, neoštećeni elementi slike će također biti izmijenjeni pod njegovim utjecajem.Ova nepoželjna situacija se događa jer konvencionalni medijan pristupi filtriranja primjenjuju medijan operacije na svaki element slike bez obzira je li on oštećen ili ne. Zbog toga će bitni detalji slike, kao što su rubovi, koji su sastavljeni od neoštećenih elemenata slike svejedno biti podvrgnuti filtriranju, što će u konačnici rezultirati smanjenjem kvalitete slike. Kako bi se riješio ovaj problem potrebno je provesti detekciju impulsnog šuma prije samog filtriranja, pa će tako samo oni elementi slike koji budu identificirani kao „oštećeni“ biti podvrgnuti procesu filtriranja, dok će oni „neoštećeni“ ostati netaknuti. O tome će se više reći u poglavlju posvećenom adaptivnom medijan filtaru i BDND algoritmu.

3.1.1. Uklanjanje impulsnog šuma pomoću standardnog medijan filtra korištenjem programskog paketa MATLAB

U MATLAB-u već postoje ugrađene funkcije raznih filtara zbog čega je pogodan za razvoj i testiranje različitih algoritama. Kako bi se ilustrirala efikasnost medijan filtra za uklanjanje impuslnog šuma tipa sol i papar, provedeno je filtriranje impulsnog šuma sa slike Lena kojoj je šumom degradirano 30%, 50%, 70% odnosno 90% elemenata slike.

Na slici 3.2 a) s lijeve strane prikazana je izvorna test slika Lena, na slici 3.2. b) vidljiva je slika degradirana s 30% umjetnog sol&papar šuma u MATLAB-u. Na slici 3.2. c) prikazana je slika na kojoj je primjenjen standardni medijan filtar.

a) Izvorna slika b) Slika s 30% oštećenja c) Filtrirana slika

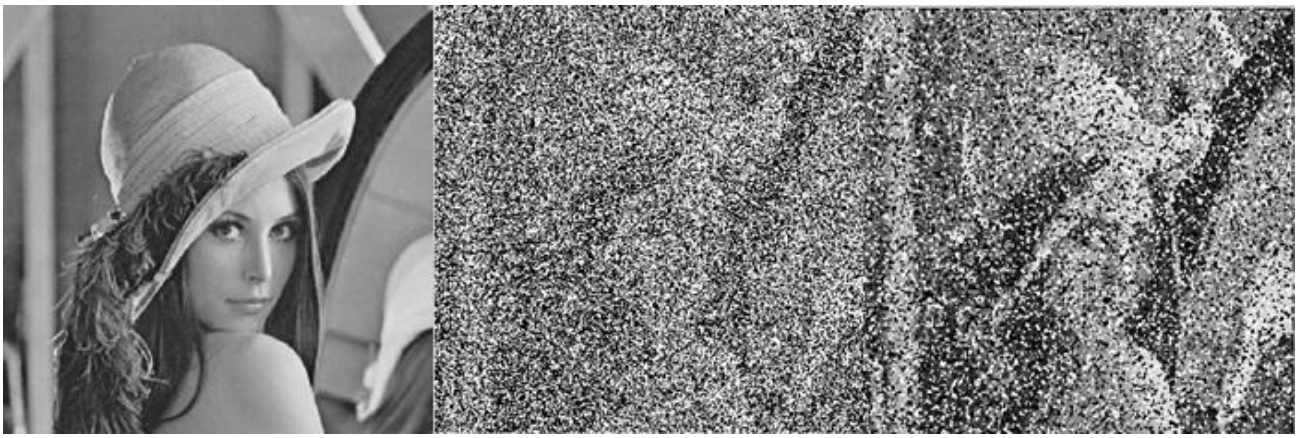
Sl. 3.2.Filtriranje Lena slike s 30% oštećenja

Na slici 3.3. a) s lijeve strane prikazana je izvorna test slika Lena, na slici 3.3. b) vidljiva je slika degradirana s 50% umjetnog sol&papar šuma u MATLAB-u. Na slici 3.3. c) prikazana je slika na kojoj je primjenjen standardni medijan filtar.

a) Izvorna slika b) Slika s 50% oštećenja c) Filtrirana slika

Sl. 3.3. Filtriranje Lena slike s 50% oštećenja

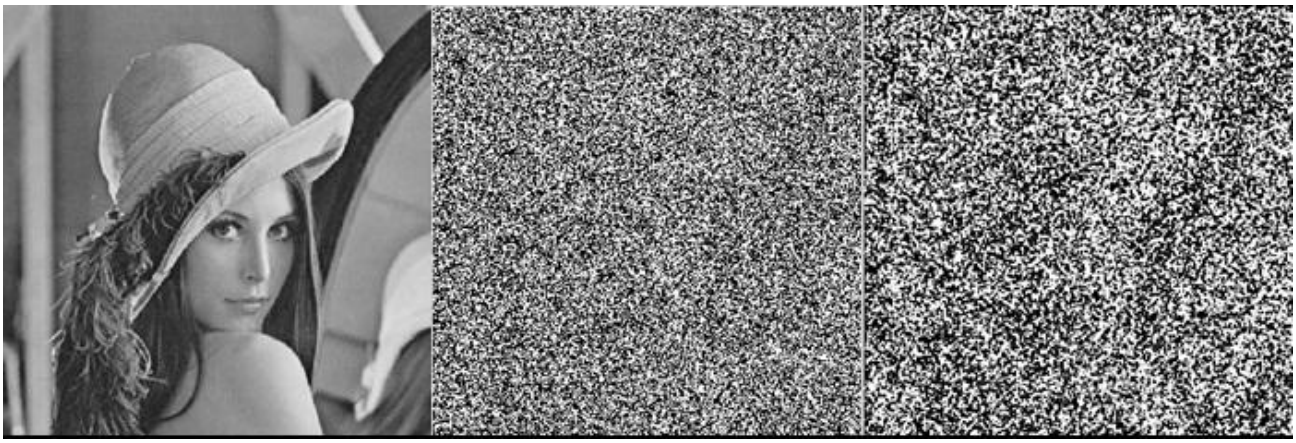
Na slici 3.4. a) s lijeve strane prikazana je izvorna test slika Lena, na slici 3.4. b) vidljiva je slika degradirana s 70% umjetnog sol&papar šuma u MATLAB-u. Na slici 3.4. c) prikazana je slika na kojoj je primjenjen standardni medijan filtar.



a) Izvorna slika b) Slika s 70% oštećenja c) Filtrirana slika

Sl. 3.4.Filtriranje Lena slike s 70% oštećenja

Na slici 3.5. a) s lijeve strane prikazana je izvorna test slika Lena, na slici 3.5. b) vidljiva je slika degradirana s 90% umjetnog sol&papar šuma u MATLAB-u. Na slici 3.5. c) prikazana je slika na kojoj je primjenjen standardni medijan filtar.



a) Izvorna slika b) Slika s 90% oštećenja c) Filtrirana slika

Sl. 3.5.Filtriranje Lena slike s 90% oštećenja

Medijan filtar dobro uklanja šum koji je prisutan na slici do 30%. Rezultati filtriranja pokazuju kako njegova učinkovitost nakon toga opada. Slika s 50% oštećenja nakon filtriranja gubi dosta detalja i rubova, ali se izvorna slika još uvijek može relativno dobro vidjeti, dok slike koje su oštećene sa 70 i 90% ni nakon filtriranja više nisu prepoznatljive.

3.2. Selektivni adaptivni medijan filtar

Zbog navedenih problema s kojima se susreće standardni medijan filtar prilikom uklanjanja šuma iz slika razvijene su razne poboljšane inačice ove metode. Jedna od tih inačica je i adaptivni medijan filtar (AMF), [5].

Koncentracija impulsnog šuma u slikama varira, jer je impulsni šum zapravo šum koji se pojavljuje nasumično. Zbog toga postoje mjesta slike sa visokom razinom oštećenja, te mjesta s niskom razinom oštećenja. Za efikasan proces filtriranja, filtar s većom konvolucijskom maskom bi trebao biti primijenjen na mjesta s visokom razinom oštećenja. Suprotno tome, filtar s manjom maskom bi trebao biti primijenjen na mjesta s niskom razinom oštećenja. Sukladno tome predstavljeni su brojni radovi koji su ponudili metode koje su u mogućnosti prilagoditi veličinu filtra s obzirom na razine oštećenja slike. Upravo iz tog razloga, jer se može prilagođavati razinama oštećenja, ovaj model medijan filtra se naziva adaptivni medijan filtar. Uobičajeno je da je veličina filtra na početku svakog procesiranja postavljena na 3×3. Veličina filtra se poslije toga postepeno širi sve dok ne zadovolji kriterije.Adaptivni medijan filtar obavlja prostornu obradu kako bi utvrdio koji su elementi slike zahvaćeni šumom. On klasificira određene elemente slike kao šum tako da prvo svaki element slike usporedi s njegovim susjedstvom. Element slike koji se znatno razlikuje od većine svojih

susjedstva ili pak nije strukturalno usklađen s onima kojima je sličan, klasificira se kao impulsni šum. Time je ovaj filtar selektivan. Elementi slike klasificirani kao impulsni šum bit će zamijenjeni s medijan vrijednostima elemenata slike u susjedstvu koji su prošli prethodno naveden test. [6] Koraci algoritma adaptivnog medijan filtara su slijedeći:

1.) A1 = Zmed – Zmin

A2 = Zmed – Zmax

ako je A1 > 0 i A2 < 0, ići na korak 2)

ako nije povećati prozor

ako je veličina prozora < Smax, ponoviti korak 1)

2.) B1 = Zxy – Zmin

B2 = Zxy – Zmax

ako je B1 > 0 i B2 < 0, izlaz je Zxy

ako nije izlaz je Zmed

gdje je Zmin minimalna vrijednost sive u promatranom susjedstvu, Zmax maksimalna vrijednost sive u promatranom susjedstvu. Zmed je medijan sivih vrijednosti u susjedstvu, Zxy je razina sivih na koordinatama (x, y) i Smax je maksimalna dopuštena veličina prozora.

Objašnjenje algoritma:

1) Ako je Zmin < Zmed < Zmax, onda

Zmed nije šum
(1) ići na korak 2 provjeriti je li Zxy šum
inače
Zmed je šum
(1) prozor se povećava i
(2) korak 1) se ponavlja dok
(a) Zmed nije šum, ići na korak 2) ili
(b) Smax je dosegnut: izlaz je Zxy

2) Ako je Zmin < Zxy < Zmax, onda

Zxy nije šum

(1) izlaz je Zxy (poremećaj je smanjen)

Inače

Zxy = Zmin ili Zxy = Zmax

(2) izlaz je Zmed

Zmed nije šum

Iako je adaptivni medijan filtar dobar u obnovi oštećenih slika, ovi filtri obično zahtijevaju dugo računalno vrijeme kada su slike vrlo oštećene. Brzina izvođenja obrade slika, odnosno uklanjanja šumova može se povećati implementiranjem koda u C programski jezik što će biti objašnjeno u sljedećim poglavljima.

3.2.1. Uklanjanje impulsnog šuma pomoću adaptivnog medijan filtra primjenom programskog paketa MATLAB

MATLAB kod koji je u ovom diplomskom radu razvijen za potrebe uklanjanja šuma iz slike pomoću adaptivnog medijan filtra s detekcijom šuma, napravljen je primjenom verzije MATLAB R2010a.

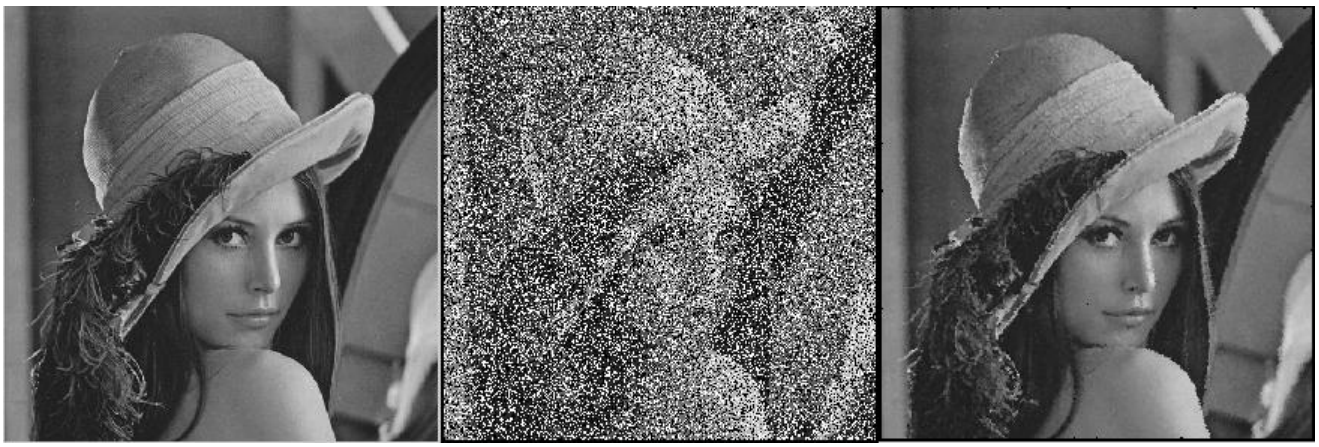
Sam kod se sastoji od 62 linije koda (Prilog). Veličina korištenog prozora adaptivnog medijan filtra je 3x3. Cilj je bio omogućiti korisniku da na samom početku pokretanja koda opcijom *Run* dobije mogućnost odabrati bilo koju sliku sa svog osobnog računala na koju želi primjeniti kod. Nakon što je korisnik odabrao željenu sliku sa svog računala u Command Window-u se pojavi poruka ' Enter the percentage of salt&pepper noise: ' gdje korisnik mora upisati kolikim postotkom, u intervalu od 1 do 100, sol&papar šuma želi degradirati izvornu sliku. Duljina trajanja uklanjanja šuma primjenom napisanog MATLAB koda ovisi o postotku šuma kojim će se degradirati slika. Na primjer ako je slika veličine 512×512 elemenata slike degradirana sa samo 5% sol&papar šuma, MATLAB kod će se izvoditi 30 sekundi, dok će se to vrijeme kod slika degradiranih s 95% povećati na otprilike 1 minutu. Iako je u spomenutom testiranju radi usporedbe trajanja izvođenja korištena ista slika, duljina izvođenja programa ovisiti će i o veličini korištenih slika. Kod testiranja koristile su se standardna test slike Lena u png formatu, visine 512 elemenata slike, širine 512 elemenata slike te veličine 443 KB. Navedena slika je u boji, ali je u svrhu testiranja pretvorena u monokromatsku sliku (*grayscale image*)

Na slici 3.6. a) s lijeve strane prikazana je izvorna test slika Lena, na slici 3.6. b) vidljiva je slika degradirana s 30% umjetnog sol&papar šuma u MATLAB-u. Na slici 3.6. c) prikazana je slika na kojoj je primjenjen adaptivni medijan filtar.

a) Izvorna slika b) Slika s 30% oštećenja c) Filtrirana slika

Sl. 3.6. Filtriranje Lena slike s 30% oštećenja

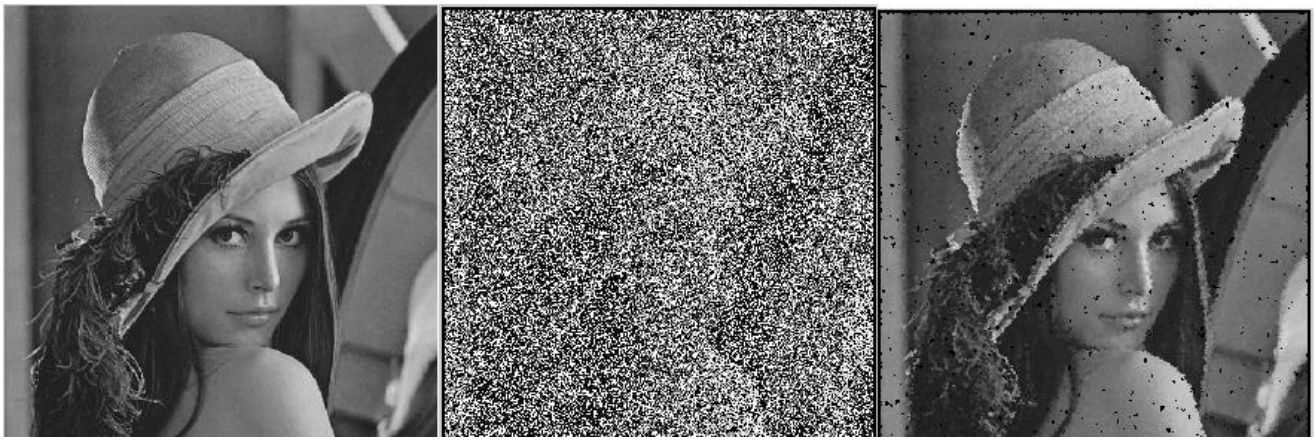
Na slici 3.7. a) s lijeve strane prikazana je izvorna test slika Lena, na slici 3.7. b) vidljiva je slika degradirana s 50% umjetnog sol&papar šuma u MATLAB-u. Na slici 3.7. c) prikazana je slika na kojoj je primjenjen adaptivni medijan filtar.



a) Izvorna slika b) Slika s 50% oštećenja c) Filtrirana slika

Sl. 3.7.Filtriranje Lena slike s 50% oštećenja

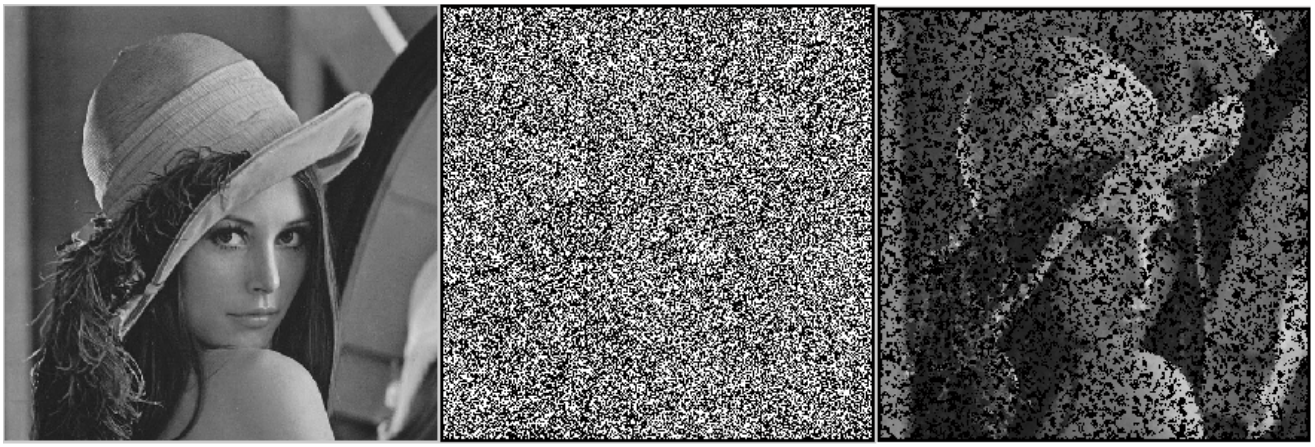
Na slici 3.8. a) s lijeve strane prikazana je izvorna test slika Lena, na slici 3.8. b) vidljiva je slika degradirana s 70% umjetnog sol&papar šuma u MATLAB-u. Na slici 3.8. c) prikazana je slika na kojoj je primjenjen adaptivni medijan filtar.



a) Izvorna slika b) Slika s 70% oštećenja c) Filtrirana slika

Sl. 3.8.Filtriranje Lena slike s 70% oštećenja

Na slici 3.9. a) s lijeve strane prikazana je izvorna test slika Lena, na slici 3.9. b) vidljiva je slika degradirana s 90% umjetnog sol&papar šuma u MATLAB-u. Na slici 3.9. c) prikazana je slika na kojoj je primjenjen adaptivni medijan filtar.



a) Izvorna slika b) Slika s 90% oštećenja c) Filtrirana slika

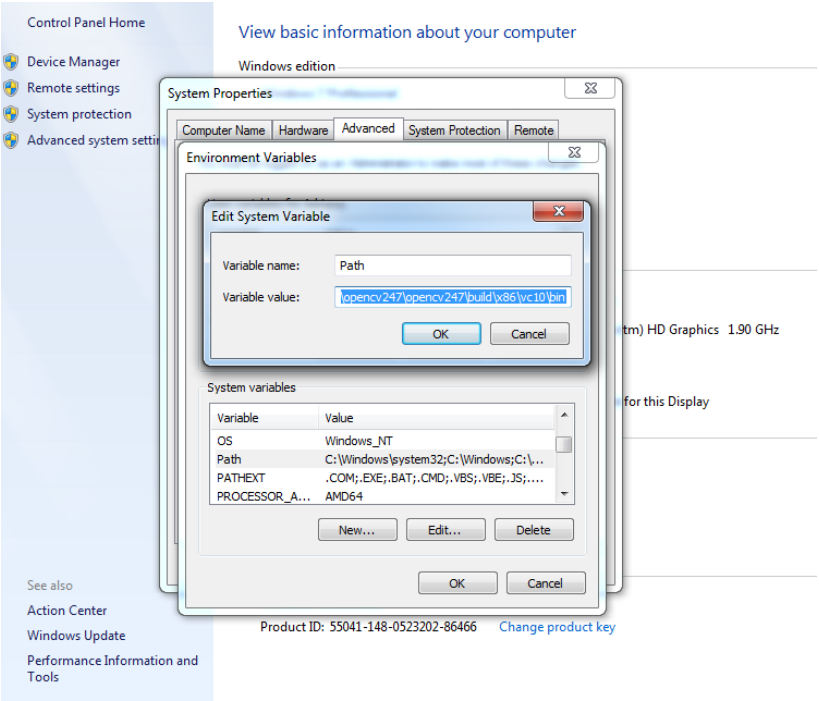
Sl. 3.9.Filtriranje Lena slike s 90% oštećenja

Za razliku od standardnog medijan filtra gdje učinkovitost uklanjanja šuma opada već nakon prvog dijela testiranja gdje je šum prisutan na 30% slike, rezultati ovog testiranja pokazuju kako je adaptivni medijan filtar vrlo učinkovit u uklanjanju do otprilike 60% šuma, nakon čega njegova učinkovitost znatno opada. Na posljednjoj slici 3.8 prikazani su razultati filtriranja kod oštećenja slike od čak 90%. Može se vidjeti da nakon filtriranja ostaje velika količina šuma te da je kvaliteta slike vrlo niska.

3.2.2. Uklanjanje impulsnog šuma pomoću adaptivnog medijan filtra primjenom C++

Za kreiranje C++ koda korišten je Visual Studio 2010. Uz Visual Studio 2010 korištena je i biblioteka OpenCV, koja se koristi kod programa za obradu slike, računalni vid i sličnih. Korištena verzija je OpenCV 2.4.7. Biblioteka OpenCv nema neku posebnu instalaciju, već kada se otvori Visual Studio projekt, treba povezati OpenCV datoteke s projektom, [8]. Za uspješan rad programa bilo je potrebno deinstalirati sve ostale verzije progama Visual Studio.

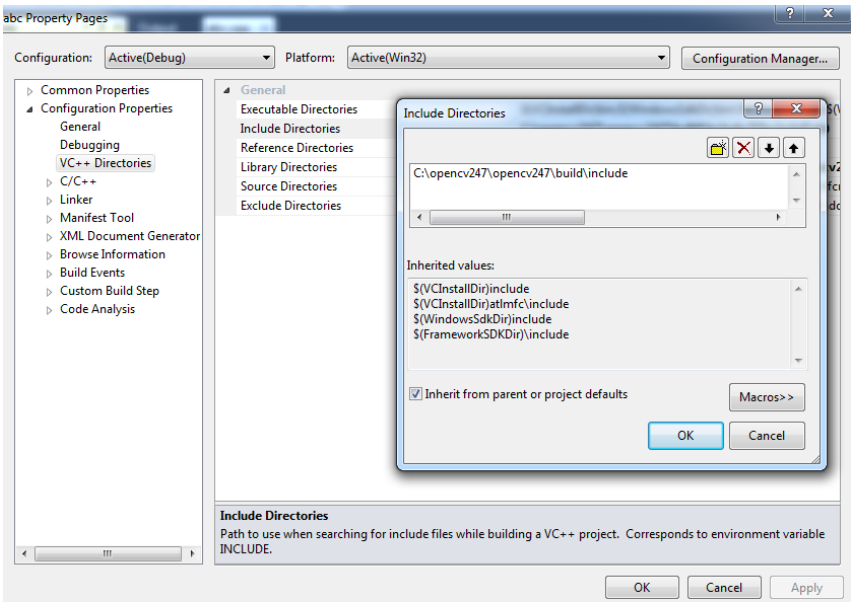
Na slici 3.10 prikazan je prozor za postavljanje konfiguracije operacijskog sustava. Nakon što se desnim klikom na *My Computer* odabere *Properties*, dobije se mogućnost s lijeve strane odabrati *Advanced system settings*. Otvara se prozor gdje se odabire opcija *Environment Variables*, nakon čega se pod *System variables* vrijednosti varijable naziva *Path* dodaje nastavak C:\opencv247\opencv247\build\x86\vc10\bin



Sl. 3.10. Konfiguracija operacijskog sustava

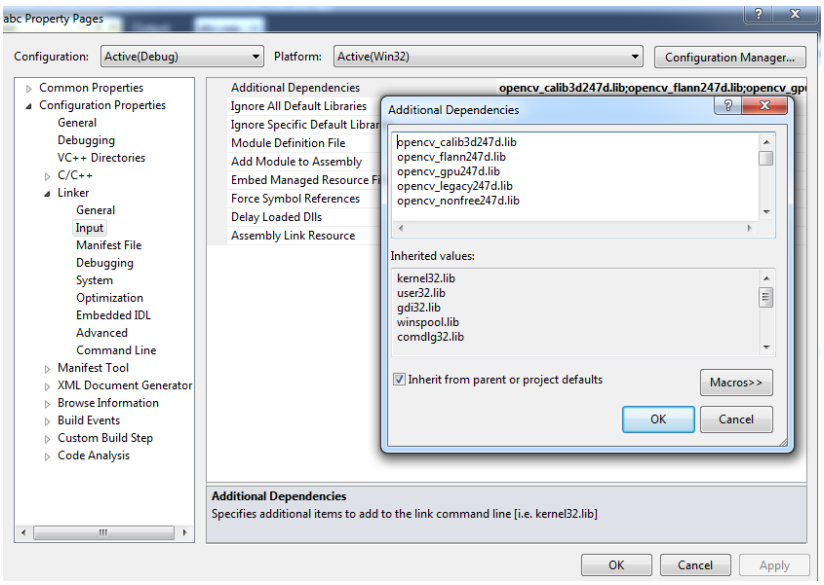
Nakon što je u Visual Studiju 2010, kreiran projekt te napisan cjelokupni kod potrebno je povezati OpenCv biblioteke s Visual Studiom. To se radi na način da se desnim klikom na projekt u prozoru Solution Explorer odabere Properties. Ondje se klikom na opciju VC++ Directories otvara prozor gdje se pod Include Directories putem opcije Browse pronalazi i upisuje C:\opencv247\openc247\build\include. Isto to je potrebno napraviti i za Library Directories, ali onda

se putem opcije Browse upisuje C:\opencv247\opencv247\build\x86\vc10\bin i C:\opencv247\opencv247\build\x86\vc10\bin. Navedeni postupak prikazan je na slici 3.11.



Sl. 3.11. Postupak povezivanja OpenCv biblioteka s projektom

Posljednji korak u povezivanju OpenCv datoteka s projektom prikazan je na slici 3.12. Odabirom opcija Linker-> Input otvara se prozor gdje se u praznom prostoru moraju upisati potrebne biblioteke.



Sl. 3.12.Postupak povezivanja OpenCv biblioteka s projektom

Dio programa koji omogućava dodavanje sol i papar šuma u sliku preuzet je od [9], a linije koda su sljedeće

```
randu(Inp,0,255); // generira objekt naizmjeničnim vrijednostima
Mat black = Inp < 40; // dodavanje papar šuma
Mat white = Inp > 215;// dodavanje sol šuma
```

Na slici 3.13 prikazan je rezultat uklanjanja sol&papar šuma pomoću adaptivnog medijan filtara i C++ programskog jezika, gdje su vrijednosti crne (papar) postavljene na <40, a vrijednosti bijele (sol) na više od 215. Postavljene vrijednosti odgovaraju postotku zasićenosti slike šumom od 31%.

a) Slika s 31% šuma b) Filtrirana slika

Sl. 3.13.Test slika Lena s 31% šuma i filtrirana slika

3.3. Detekcija šuma pomoću BDND algoritma

Jedan od vrlo preciznih algoritama za detekciju šuma je BDND (*Boundary Discriminative Noise Detection*) algoritam, [10]. koji se može nositi i sa slikama koje su oštećene i do 90%. Navedeni BDND algoritam primjenjuje se na svaki element oštećene slike pojedinačno kako bi utvrdio je li on oštećen ili nije oštećen. Nakon primjene ovog algoritma na cijelu sliku treba se formirati dvodimenzionalna binarna karta gdje „0“ predstavlja poziciju neoštećenog piksela, a „1“ predstavlja poziciju oštećenih piskela. Kako bi se to ostvarilo svi elementi slike koji se nalaze unutar predefiniranog prozora čiji je centar promatrani element biti će podijeljeni u tri skupine te se moraju odrediti dvije granice b_1 i b_2. Svaki element slike $x_{i,j}$ koji se uzme u obzir i za kojeg vrijedi da je $0 \leq x_{i,j} \leq$ b1, biti će dodijeljen grupi niskog intenziteta, ako je $b_1 < x_{i,j} \leq b_2$ biti će dodijeljen grupi srednjeg intenziteta, dok će za $b_2 < x_{i,j} \leq 255$, piksel biti dodijeljen grupi visokog intenziteta. Ako piksel $x_{i,j}$ pripada grupi srednjeg intenziteta, taj isti piksel će se smatrati neoštećenim, jer vrijednosti intenziteta nisu niti relativno visoke niti relativno niske. Ako je drugačije od toga velika je

vjerojatnost da je piksel oštećen šumom. Naravno, točnost podjele piksela u ove tri grupe, odnosno, točnost detektiranja šuma u konačnici ovisi o tome koliko su precizno utvrđene granice b_1 i b_2.

Predloženi proces detekcije šuma sastoji se od dvije iteracije, od kojih će druga iteracije biti pozvana samo uvjetovano. U prvoj iteraciji, veliki prozor veličine 10×10 koristi se kako bi provjerio je li promatrani element slike oštećen. Ako se ispostavi da je element slike neoštećen druga iteracija će biti pozvana zbog daljnjeg ispitivanja, gdje će se koristiti prozor veličine 3×3. U sljedećim koracima navedeni su koraci algoritma ove metode.

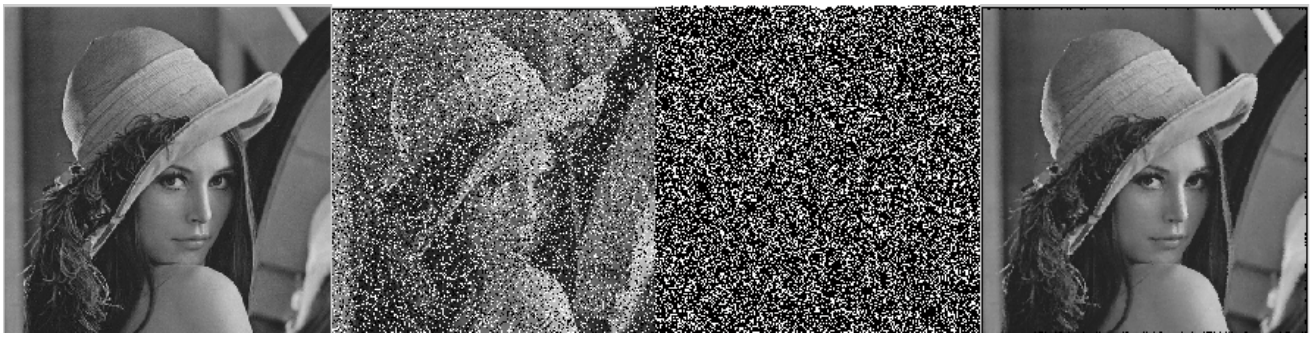
1) Postaviti prozor veličine 10×10 koji je centriran oko promatranog elemenata slike.
2) Sortirati elemente slike u prozoru prema uzlaznom redoslijedu i pronaći medijan sortiranog vektora $\mathbf{v}_0$
3) Izračunati razliku intenziteta između svakog para susjednih elemenata slike preko sortiranog vektora **v**0 i dobiti vektor razlike $\mathbf{v}_D$.
4) Za elemente slike intenziteta između 0 i medijana u $\mathbf{v}_0$ treba pronaći maksimalnu razliku intenziteta u $\mathbf{v}_D$ istog raspona i označiti odgovarajuće elemente slike u $\mathbf{v}_0$ kao granicu b1.
5) Isto tako granica b2 definirana je za elemente slike intenziteta između medijana i 255. Tako su formirana tri klastera.
6) Ako element slike pripada srednjem klasteru klasificiran je kao neoštećen element slike, i tu klasifikacija završava, inače, pozvati će se druga iteracija u nastavku.
7) Postaviti prozor veličine 3×3, koji je centriran oko trenutnog elementa slike i ponoviti korake od 2)-5)
8) Ako promatrani element slike pripada srednjem klasteru klasificiran je kao neoštećen, ako ne, klasificiran je kao oštećen, [10].

3.3.1. Uklanjanje impulsnog šuma pomoću adaptivnog medijan filtra s BDND algoritmom

Izrada ovog koda podrazumijevala je malu preinaku izvornog koda adaptivnog medijan filtra, kako bi se BDND funkcija mogla povezati s funkcijom main. Kao i kod prvog koda pokretanjem opcijom Run dobije se mogućnost odabira bilo koje slike s osobnog računala i mogućnost unosa postotka sol&papar šuma kojim se želi degradirati izvorna slika. Nakon toga program počinje s obradom slike. Upravo na ovom primjeru može se vidjeti koliko je MATLAB spor u izvođenju malo zahtijevnijih obrada slike. Izvođenje ovog koda za uklanjanje šuma iz relativno male test slike Lena

(512x512 elemenata slike) traje čak 15 minuta što je kod obrade slike neprihvatljivo visoko vrijeme. U nastavku su prikazani rezultati testiranja s 30%, 50%, 70% i 90% oštećenosti impulsnim sol&papar šumom.

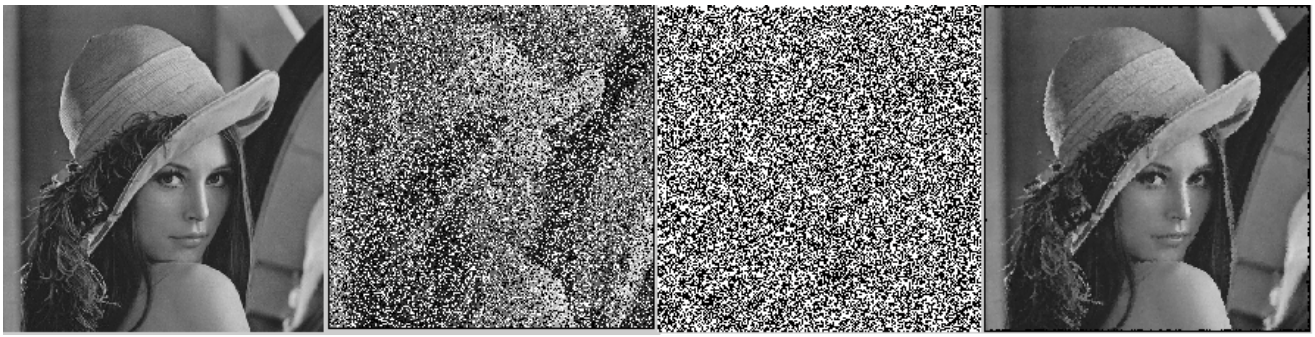
Na slici 3.14. a) s lijeve strane prikazana je izvorna test slika Lena, na slici 3.14. b) vidljiva je slika degradirana s 30% umjetnog sol&papar šuma u MATLAB-u. Na slici 3.14. c) vidljiva je BDND maska. Na slici 3.14. d) prikazana je slika na kojoj je primjenjen adaptivni medijan filtar u kombinaciji s BDND algoritmom.



a) Izvorna slika b) Slika s 30% oštećenja c) BDND maska d) Filtrirana slika

Sl. 3.14.Uklanjanje šuma BDND algoritmom Lena slike s 30% oštećenja

Na slici 3.15. a) s lijeve strane prikazana je izvorna test slika Lena, na slici 3.15. b) vidljiva je slika degradirana s 50% umjetnog sol&papar šuma u MATLAB-u. Na slici 3.15. c) vidljiva je BDND maska. Na slici 3.15. d) prikazana je slika na kojoj je primjenjen adaptivni medijan filtar u kombinaciji s BDND algoritmom.

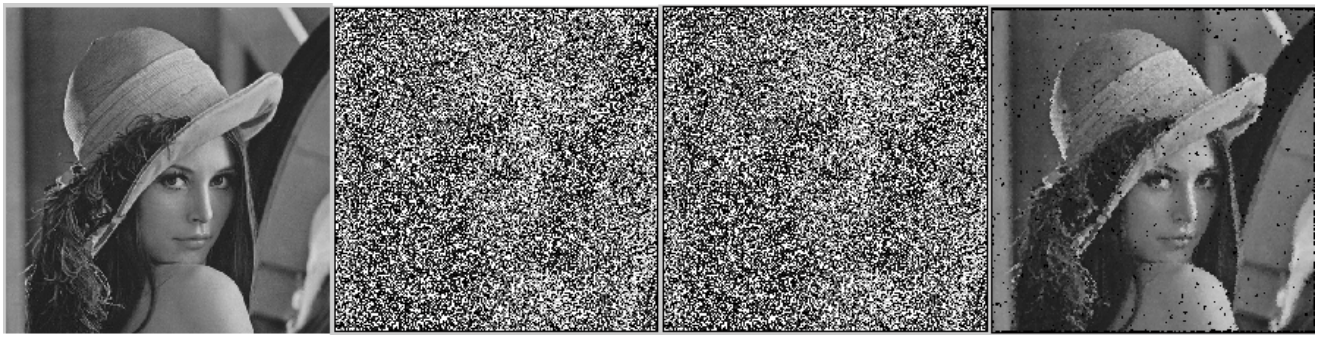


a) Izvorna slika b) Slika s 50% oštećenja c) BDND maska d) Filtrirana slika

Sl. 3.15.Uklanjanje šuma BDND algoritmom Lena slike s 50% oštećenja

Na slici 3.16. a) s lijeve strane prikazana je izvorna test slika Lena, na slici 3.16. b) vidljiva je slika degradirana s 70% umjetnog sol&papar šuma u MATLAB-u. Na slici 3.16. c) vidljiva je BDND

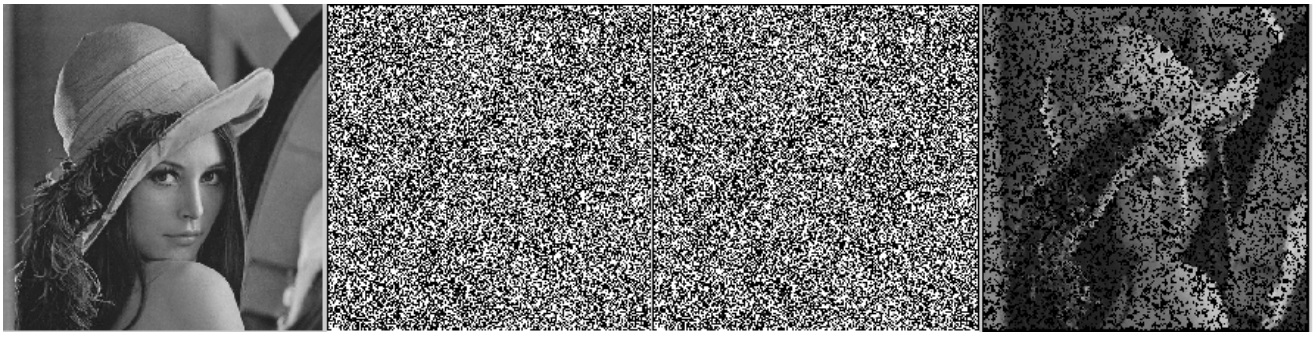
maska. Na slici 3.16. d) prikazana je slika na kojoj je primjenjen adaptivni medijan filtar u kombinaciji s BDND algoritmom.



a) Izvorna slika b) Slika s 70% oštećenja c) BDND maska d) Filtrirana slika

Sl. 3.16.Uklanjanje šuma BDND algoritmom Lena slike s 70% oštećenja

Na slici 3.17. a) s lijeve strane prikazana je izvorna test slika Lena, na slici 3.17. b) vidljiva je slika degradirana s 90% umjetnog sol&papar šuma u MATLAB-u. Na slici 3.17. c) vidljiva je BDND maska. Na slici 3.17. d) prikazana je slika na kojoj je primjenjen adaptivni medijan filtar u kombinaciji s BDND algoritmom.



a) Izvorna slika b) Slika s 90% oštećenja c) BDND maska d) Filtrirana slika

Sl. 3.17.Uklanjanje šuma BDND algoritmom Lena slike s 90% oštećenja

Uspoređujući slike 3.8 (adaptivni medijan sa 70% oštećenja) i 3.16 (BDND algoritam sa 70% oštećenja), mogu se zamijetiti neka vrlo mala poboljšanja kod uklanjanja šuma primjenom adaptivnog medijan filtra u odnosu na BDND algoritam. Kad se postotak oštećenja poveća na čak 90% ako i postoji neko poboljšanje ono zbog ogromne količine šuma postaje nezamjetno. Može se zaključiti kako adaptivni medijan filtar efikasno uklanja šum kod oštećenja do 60-65%. Primjena BDND algoritma, iako donosi mala poboljšanja kod uklanjanja šuma, s obzirom na vrlo visoko vrijeme izvođenja do čak 15 minuta za vrlo male slike, nema posebno opravdanje.

3.4. Usporedba efikasnosti MF, AMF i BDND

Kako bi se najbolje i najtočnije mogle usporediti efikasnosti metoda obrađenih u ovom radu, upotrijebit će se takozvani PSNR (Peak signal-to-noise ratio). PSNR je zapravo izraz za omjer vršne snage signala i snage šuma koji utječe na sliku. Budući da mnogi signali imaju širok dinamički raspon, PSNR se obično izražava u decibelima.

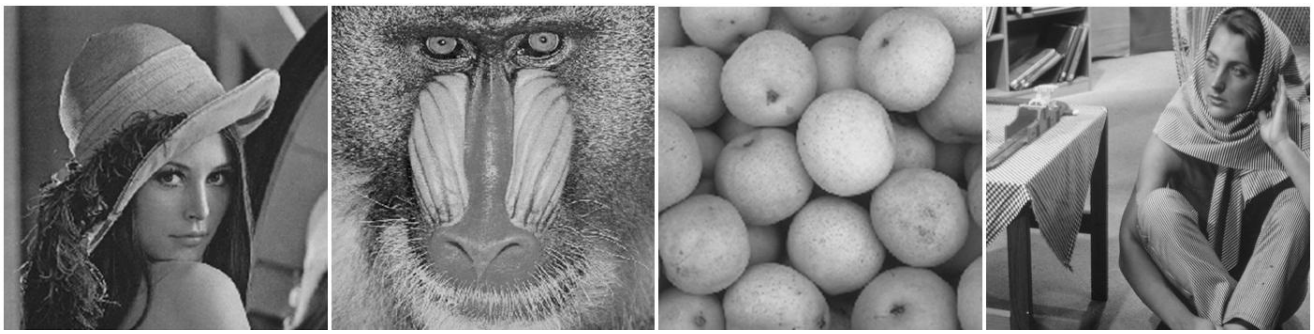
PSNR se najčešće koristi za mjerenje kvalitete rekonstrukcije. Veći PSNR općenito ukazuje na to da je rekonstrukcija slike kvalitetnija. PSNR se definira preko srednje kvadratne pogreške MSE (eng. Mean Squared Error), [11]. Za monokromatsku sliku bez šuma I(i,j) veličine m × n elemenata slike i njenu degradiranu verziju K(i,j), MSE je definiran kao:

$$\text{MSE} = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{3-2}$$

PSNR (u dB) je definiran kao:

$$\text{PSNR} = 10 \times \log \{10(\{256\}\,\text{'}2/MSE\} \tag{3-3}$$

MATLAB kodovima za standardni medijan filtar (MF), adaptivni medijan filtar (AMF), te adaptivni medijan filtar s BDND algoritmom dodane su linije koda za proračun MSE i PSNR vrijednosti. Korištene slike su Lena i Baboon, te Pears i Barbara i to s postotcima sol i papar šuma od 10, 20, 30, 40, 50, 60, 70, 80 i 90%. Upravo ta dva para slika odabrana su zbog njihovih različitih sadržaja s obzirom na količinu detalja i rubova. Dok je na Lena i Pears slikama sve jasno definirano i vidljivo kod Baboon i Barbara slika to nije slučaj, već se one sastoji od mnoštva sitnijih detalja, što utječe na kvalitetu rekonstukcije slike oštećene šumom. Na slici 3.18. prikazane su, s lijeva na desno, korištene test slike Baboon, Lena, Pears i Barbara.



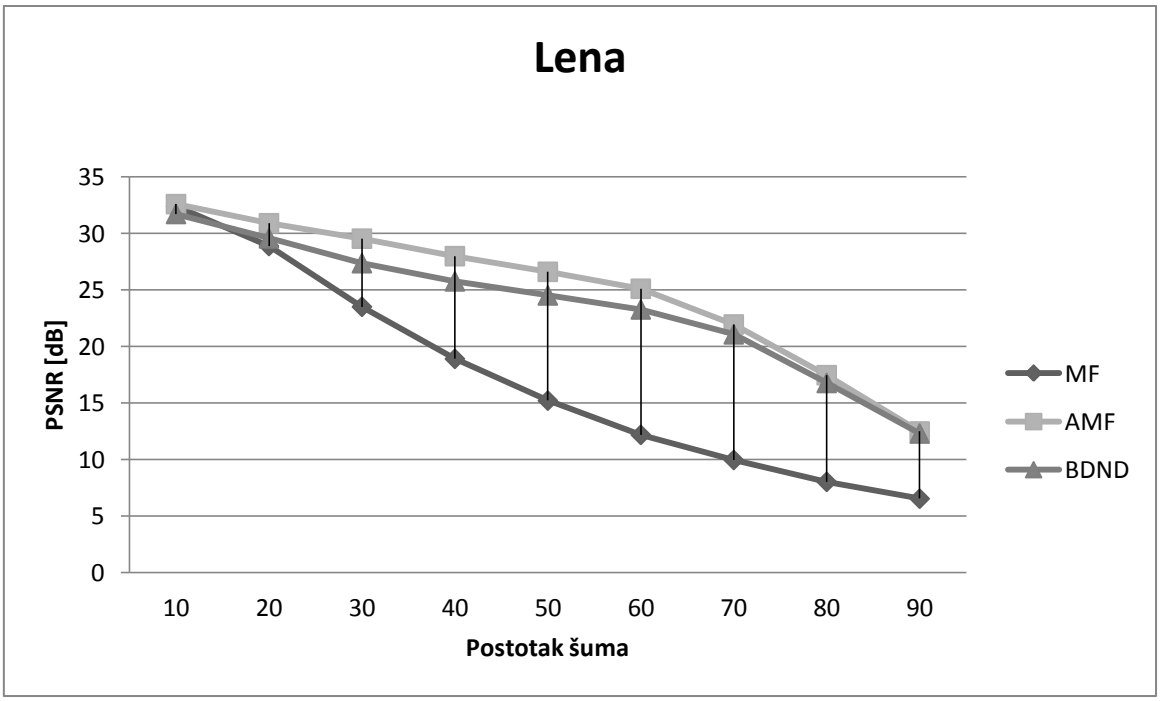
a) Lena b) Baboon c) Pears d) Barbara

Sl. 3.18. Korištene test slike

Vrijednosti PSNR za Lena test sliku nakon uklanjanja impulsnog sol i papar šuma primjenom medijan filtra, adaptivnog medijan filtra i adaptivnog medijan filtra s BDND algoritmom prikazani su u tablici 3.1. i na slici 3.19. Rezultati pokazuju kako je za uklanjanje sol i papar šuma iz slike koja ima relativno jasno definirane rubove i ne previše detalja, najviše uspjeha imao adaptivni medijan filtar. Iako i adaptivni medijan filtar u kombinaciji s BDND algoritmom pokazuje odlične rezultate, zbog duljine izvođenja od čak 15 minuta njegovo korištenje u ovom slučaju nije isplativo. Standardni medijan filtar pokazuje vrlo dobre rezultate kada je slika zasićena šumom do 20%, ali nakon toga njegova efikasnost značajno opada.

Tab.3.1.Rezultati PSNR testiranja za test sliku Lena

	Filtar	Postotak šuma								
		10%	20%	30%	40%	50%	60%	70%	80%	90%
PSNR (dB)	MF	32,38	28,88	23,50	18,90	15,21	12,18	9,96	8,03	6,53
	AMF	32,58	30,89	29,53	27,98	26,60	25,10	21,94	17,46	12.48
	BDND	31,71	29,55	27,36	25,75	24,51	23,27	21,07	16,77	12,30

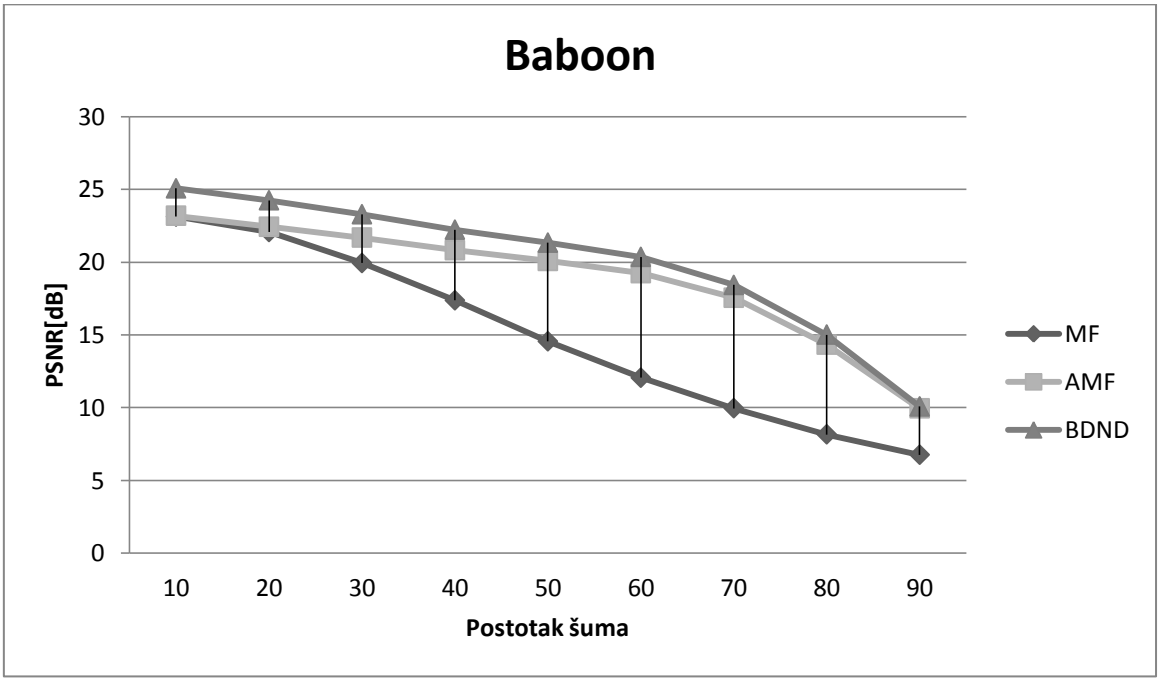


Sl.3.19. Rezultati testiranja za test sliku Lena

Rezultati testiranja za Baboon test sliku, koji su prikazani na tablici 3.2. i slici 3.20. pokazuju kako najveću učinkovitost prilikom uklanjanja sol i papar šuma iz slike ima adaptivni medijan filtar u kombinaciji s BDND algoritmom. To dovodi do zaključka kako je upravo taj filtar idealan izbor kod slika koje imaju jako puno detalja, te mnoštvo nejasnih rubova. Obični medijan filtar kao i u prošlom slučaju znatno gubi na učinkovitosti nakon zasićenosti šumom koja je veća od 20%.

Tab.3.2.Rezultati PSNR testiranja za test sliku Baboon

	Filtar	Postotak šuma								
		10%	20%	30%	40%	50%	60%	70%	80%	90%
PSNR (dB)	MF	23,13	22,07	19,94	17,37	14,56	12,05	9,96	8,16	6,76
	AMF	23,19	22,43	21,69	20,82	20,08	19,24	17,55	14,33	9,95
	BDND	25,07	24,25	23,29	22,23	21,34	20,36	18,43	15,02	10,08

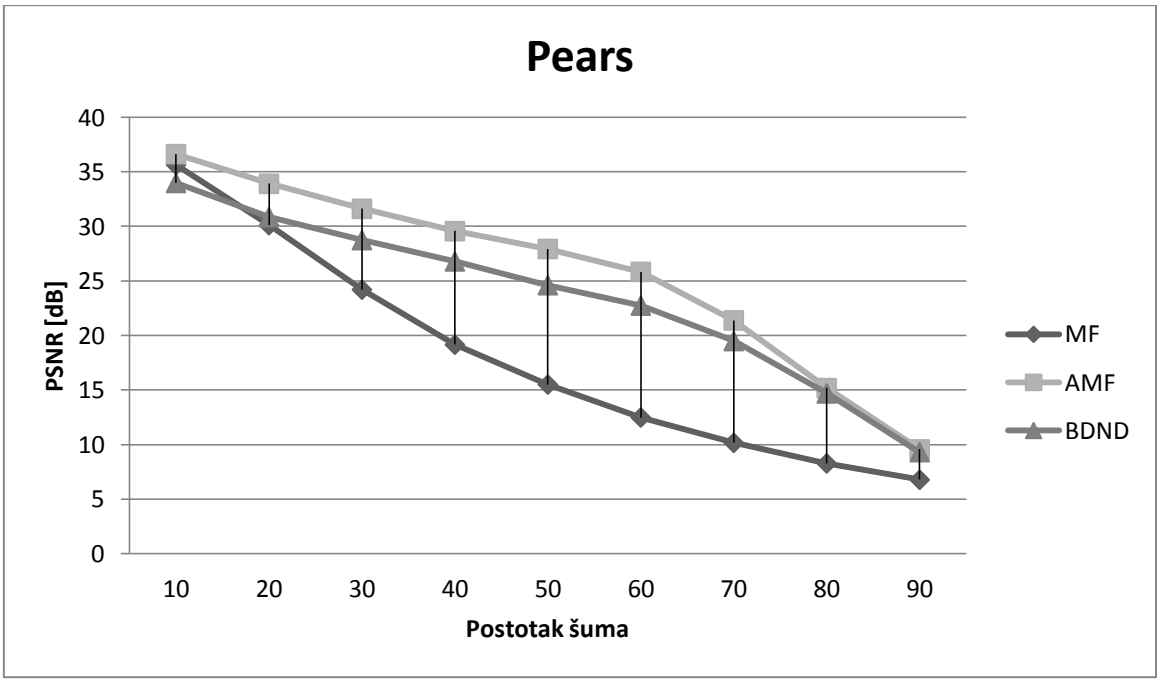


Sl.3.20.Rezultati testiranja za test sliku Baboon

Na tablici 3.3. i slici 3.21. prikazani su rezultati filtriranja gdje je korištena slika Pears koja je poprilično jednostavna bez presitnih detalja. Može se vidjeti kako je najbolje rezultate dao adaptivni medijan filtar, dok je i u ovom slučaju standardni medijan filtar podbacio nakon zasićenja šumom koji je veći od 20%.

Tab.3.3.Rezultati PSNR testiranja za test sliku Pears

	Filtar	Postotak šuma								
		10%	20%	30%	40%	50%	60%	70%	80%	90%
PSNR (dB)	MF	35,65	30,12	24,21	19,17	15,51	12,49	10,17	8,28	6,78
	AMF	36,63	33,91	31,62	29,57	27,94	25,84	21,39	15,19	9,57
	BDND	33,99	30,86	28,75	26,80	24,52	22,75	19,50	14,70	9,32

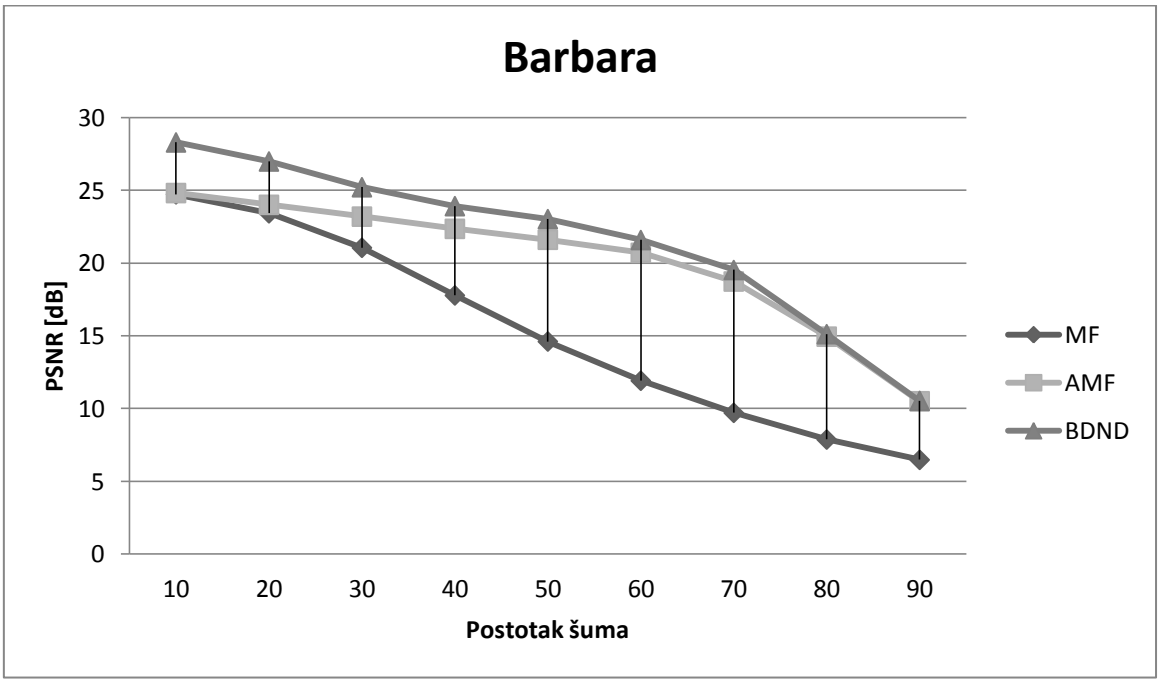


Sl.3.21.Rezultati testiranja za test sliku Pears

Na tablici 3.4. i slici 3.22. prikazani su rezultati filtriranja gdje je korištena test slika Barbara koja je za razliku od test slike Pears složenija, s puno više sitnih detalja. Ovaj dio testiranja potvrdio je ono što je prvi dio testiranja pokazao, a to je da kod složenijih slika koje se sastoje od mnoštva detalja puno veći uspjeh u uklanjanju šuma ima adaptivni medijan filtar u kombinaciji s BDND algoritmom. Standardni medijan filtar kao i u prethodna tri slučaja ima najlošije rezultate.

Tab.3.4.Rezultati PSNR testiranja za test sliku Barbara

	Filtar	Postotak šuma								
		10%	20%	30%	40%	50%	60%	70%	80%	90%
PSNR (dB)	MF	24,73	23,43	21,06	17,78	14,62	11,99	9,17	7,88	6,48
	AMF	24,82	24,01	23,20	22,36	21,61	20,71	18,74	14,95	10,51
	BDND	28,30	26,99	25,22	23,92	23,03	21,59	19,53	15,12	10,54



Sl.3.22.Rezultati testiranja za test sliku Barbara

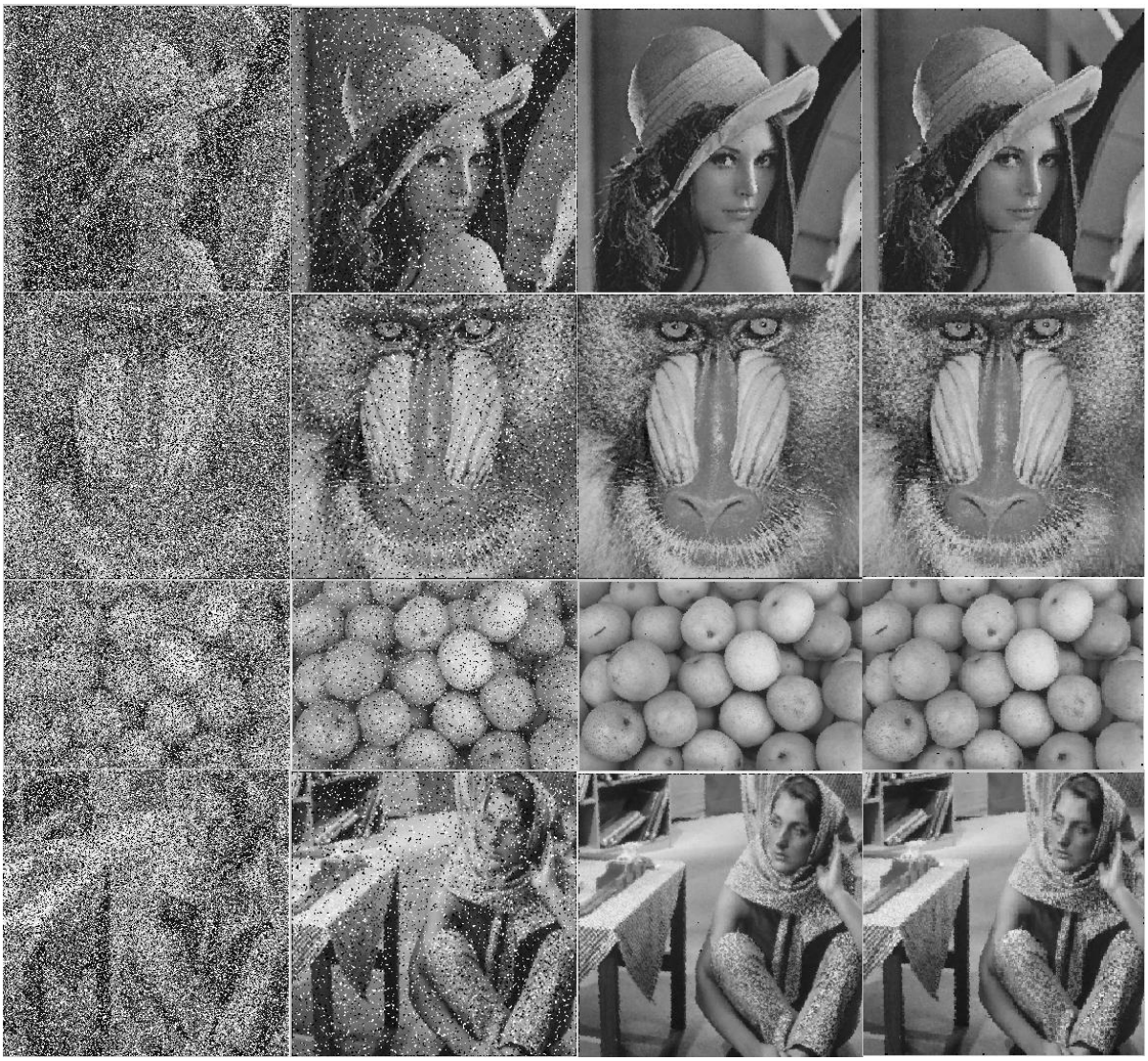
Na slici 3.23. a) prikazane su četiri test slike koje su degradirane s 20% sol i papar šuma. Filtriranje je napravljeno pomoću MF, AMF i BDND. Slika 3.23. b) prikazuje slike nakon filtriranja standardnim medijan filtrom. Na slici 3.23. c) prikazane su slike filtrirane adaptivnim medijanom filtrom, dok posljednja 3.23. d) prikazuje slike filtrirane adaptivnim medijan filtrom u kombinaciji s BDND algoritmom.

a) Slike s 20% šuma b) Slike s MF c) Slike s AMF d) Slike s BDND

Sl.3.23.Rezultati filtriranja slika s 20% šuma pomoću MF, AMF i BDND

Na slici 3.24. a) prikazane su četiri test slike koje su degradirane s 50% sol i papar šuma. Filtriranje je napravljeno pomoću MF, AMF i BDND. Slika 3.24. b) prikazuje slike nakon filtriranja standardnim medijan filtrom. Na slici 3.24. c) prikazane su slike filtrirane adaptivnim medijanom

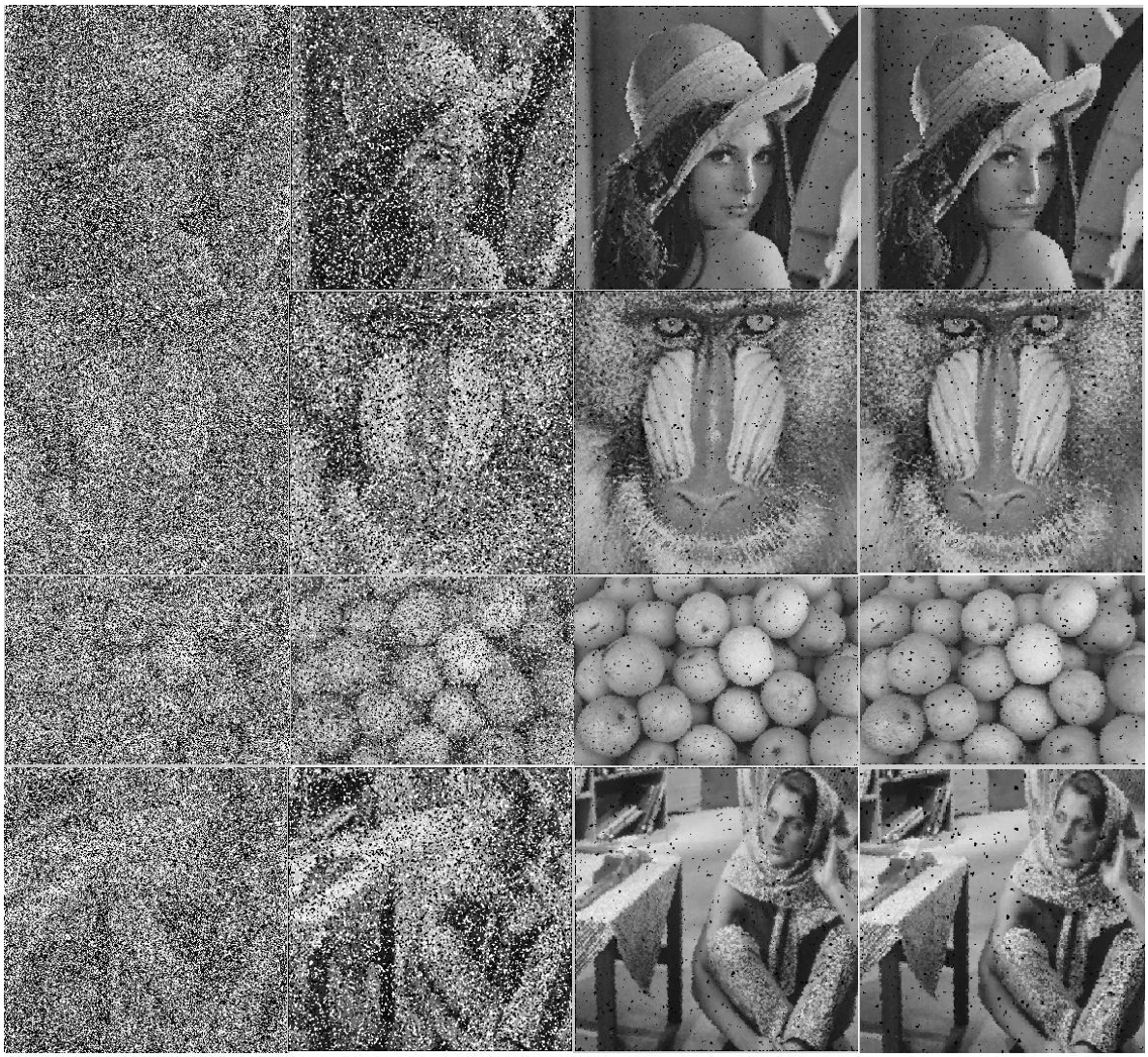
filtrom, dok posljednja 3.24. d) prikazuje slike filtrirane adaptivnim medijan filtrom u kombinaciji s BDND algoritmom.



a) Slike s 50% šuma b) Slike s MF c) Slike s AMF d) Slike s BDND

Sl.3.24.Rezultati filtriranja slika s 50% šuma pomoću MF, AMF i BDND

Na slici 3.25. a) prikazane su četiri test slike koje su degradirane s 70% sol i papar šuma. Filtriranje je napravljeno pomoću MF, AMF i BDND. Slika 3.25. b) prikazuje slike nakon filtriranja standardnim medijan filtrom. Na slici 3.25. c) prikazane su slike filtrirane adaptivnim medijanom filtrom, dok posljednja 3.25. d) prikazuje slike filtrirane adaptivnim medijan filtrom u kombinaciji s BDND algoritmom.



a) Slike s 70% šuma b) Slike s MF c) Slike s AMF d) Slike s BDND

Sl.3.25.Rezultati filtriranja slika s 70% šuma pomoću MF, AMF i BDND

Na slici 3.23. četiri test slike oštećene su relativno malim postotkom sol i papar šuma, od 20%. Rezultati filtriranja su zadovoljavajući, iako se i kod ovog malog postotka šuma može primjetiti kako standardni medijan ima najlošije rezultate, jer su na slikama filtriranim njime vidljivi ostaci čestica sol i papar šuma. Na slikama na kojima su primjenjeni AMF i BDND nema tragova prethodno dodanog šuma. Na idućoj slici, 3.24., test slike oštećene su postotkom šuma od 50%. Medijan filtar ponovno ima vidljivo najlošije rezultate, dok su slike na kojima je primjenjen AMF i BDND uspješno filtrirane. Detalji su kod njih ostali očuvani, iako se može primjetiti kako su Lena i Pears slike nakon filtriranja BDND-om ipak nešto zamućenije nego što je to slučaj nakon filtriranja AMF-om. Na posljednjoj slici, 3.25., test slike oštećene su sa 70% sol i papar šuma. Vidljivo je kako kod ovog postotka šuma MF filtar nije primjenjiv jer detalji slika više nisu očuvani, mogu se tek vrlo slabo

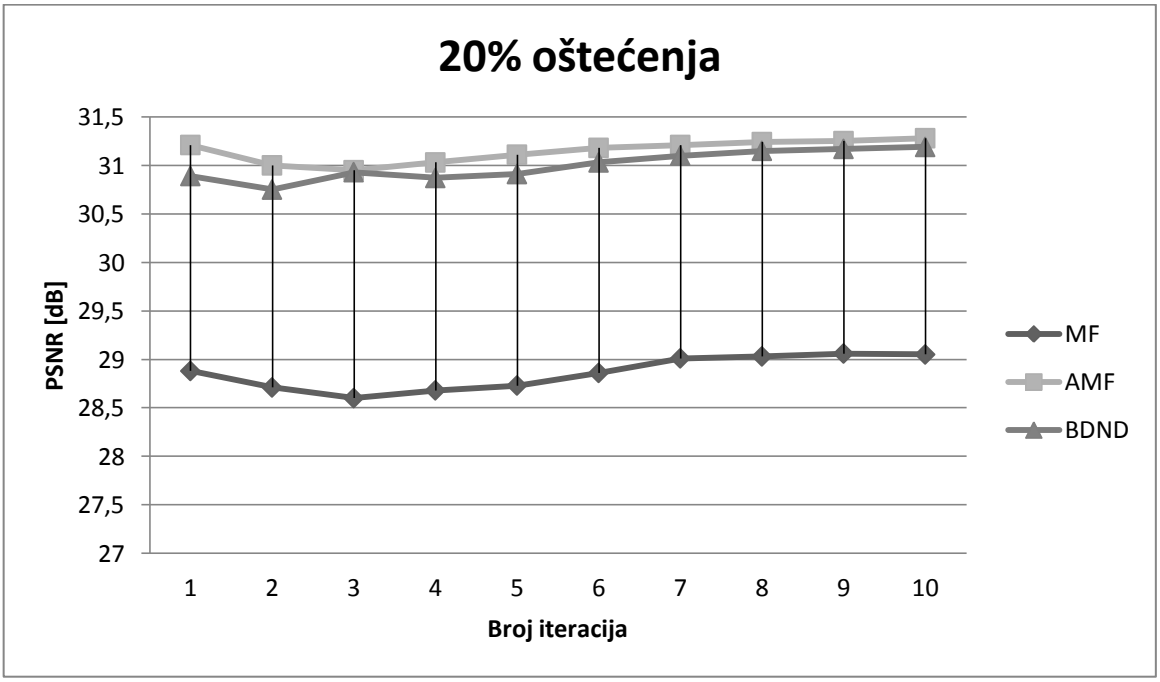
razaznati izvorne slike. S obzirom na veliki postotak šuma rezultati filtriranja AMF-om i BDND-om su zadovoljavajući, ali samo gledajući filtrirane slike ne može se doći do zaključka koji je od dva filtra učinkovitiji u uklanjanju šuma

3.4.1. Iterativni postupak uklanjanja šuma

Kako bi se provjerila mogućnost poboljšanja kvalitete slike iterativnim postupkom filtriranja impulsnog šuma, provedeno je filtriranje s MF, AMF i BDND filtrima za sliku Lena, oštećenu s 20%, 50% i 70% šuma. U tablicama 3.5. do 3.7. i na slikama 3.26. do 3.28. prikazani su rezultati PSNR primjenom 1,2,3,4,5,6,7,8,9 i 10 iteracija.

Tab.3.5. Primjena MF, AMF i BDND na sliku Lena s 20% oštećenja

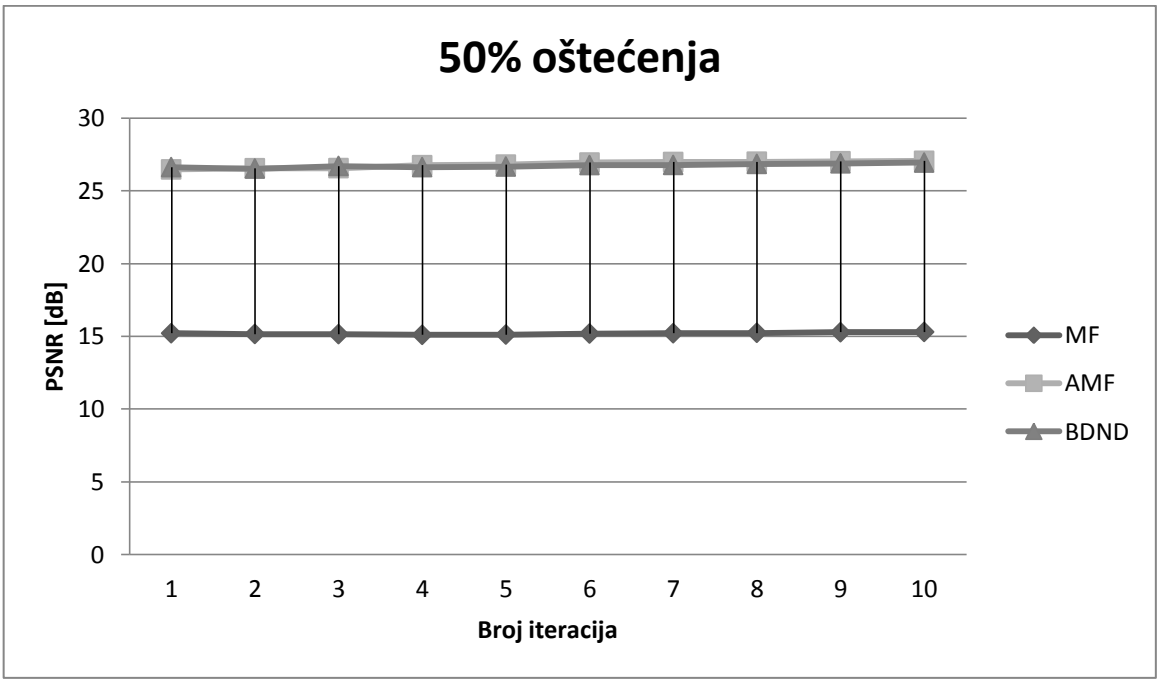
	Filtar	Broj iteracija									
		1	2	3	4	5	6	7	8	9	10
PSNR (dB)	MF	28,88	28,71	28,60	28,68	28,73	28,86	29,01	29,03	29,06	29,05
	AMF	31,21	31,00	30,95	31,03	31,11	31,18	31,21	31,24	31,25	31,28
	BDND	30,89	30,75	30,93	30,87	30,91	31,03	31,10	31,25	31,17	31,19



Sl.3.26. Ovisnost PSNR-a o broju iteracija za 20% oštećenja

Tab.3.6. Primjena MF, AMF i BDND na sliku Lena s 50% oštećenja

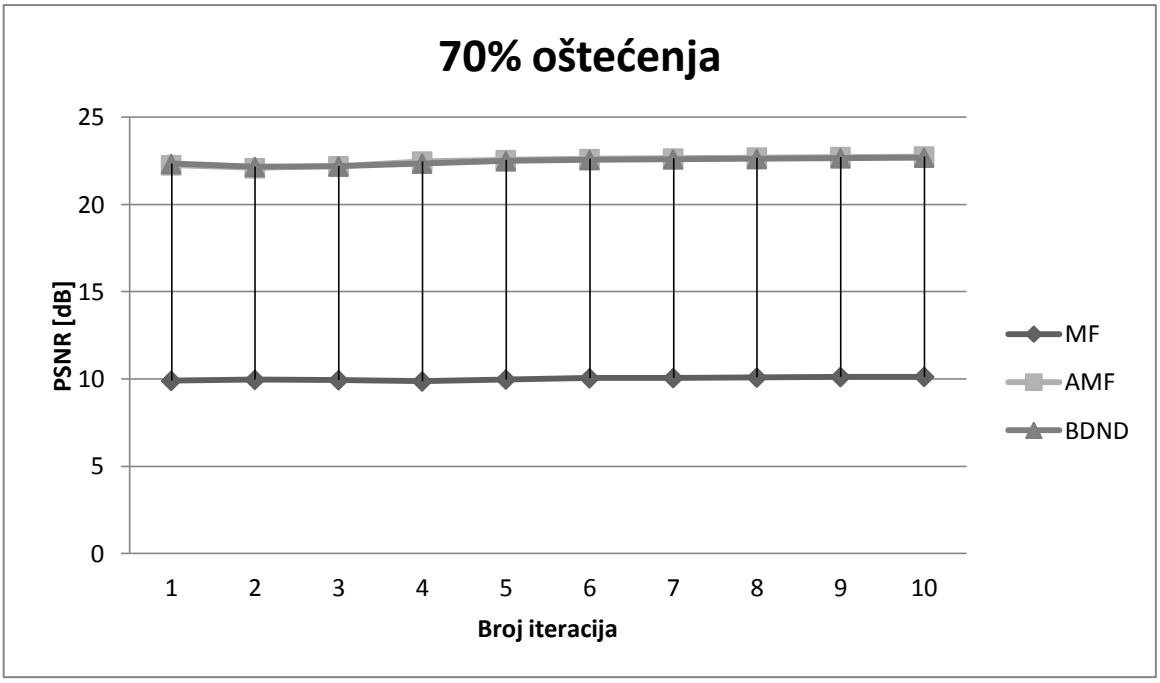
	Filtar	Broj iteracija									
		1	2	3	4	5	6	7	8	9	10
PSNR (dB)	MF	15,21	15,16	15,15	15,10	15,12	15,19	15,21	15,24	15,28	15,30
	AMF	26,47	26,55	26,56	26,76	26,80	26,94	26,98	27,00	27,03	27,06
	BDND	26,63	26,53	26,71	26,63	26,67	26,77	26,78	26,85	26,89	26,94



Sl.3.27. Ovisnost PSNR-a o broju iteracija za 50% oštećenja

Tab.3.7. Primjena MF, AMF i BDND na sliku Lena s 70% oštećenja

	Filtar	Broj iteracija									
		1	2	3	4	5	6	7	8	9	10
PSNR (dB)	MF	9,90	9,96	9,94	9,87	9,98	10,05	10,06	10,09	10,12	10,13
	AMF	22,25	22,08	22,19	22,46	22,54	22,60	22,63	22,69	22,72	22,75
	BDND	22,33	22,16	22,18	22,35	22,48	22,55	22,59	22,61	22,65	22,69



Sl.3.28. Ovisnost PSNR-a o broju iteracija za 70% oštećenja

Iterativni postupak uklanjanja šuma proveden je na slici Lena uz oštećenja od 20, 50 i 70%, primjenom standardnog medijan filtra, adaptivnog medijan filtra i adaptivnog medijana u kombinaciji s BDND algoritmom. Testiranje ovakvog postupka ograničeno je samo na navedenu sliku i postotke zbog predugog izvođenja BDND algoritma u MATLAB-u. Može se primjetiti kako PSNR niti u jednoj od iteracije nije isti. Vidljivo je i kako nakon četvrte iteracije PSNR raste u većini slučajeva. Jedina iznimka koja se može vidjeti je PSNR kod medijan filtra i to na slici s 20% oštećenja. Ovo testiranje ograničeno je na samo deset iteracija zbog vrlo dugog izvođenja BDND algoritma, ali može se zaključiti kako će svakom novom iteracijom PSNR biti sve veći. To znači da se iterativnim postupkom uklanjanja šuma kvaliteta rekonstrukcije slika povećava.

3.5. Usporedba složenosti MF, AMF i BDND algoritama

Najjednostavniji od predloženih algoritama je medijan filtar. Njegov algoritam sastoji se od toga da postavi prozor stalne veličine, promatranom elementu slike odredi medijan vrijednosti susjednih elemenata slike te na kraju taj isti element slike zamijeni medijanom. Drugi po složenosti algoritam je adaptivni medijan filtar. On na početku izvođenja kao i medijan ima prozor veličine 3×3. Za razliku od medijana AMF će prvo napraviti provjeru elemenata slike kako bi utvrdio koji su od njih zahvaćeni šumom, te će po potrebi povećavati prozor dok on ne dostigne svoju maksimalnu veličinu ili dok ne otkrije šum. Najsloženiji od ova tri algoritama je adaptivni medijan filtar u kombinaciji s BDND algoritmom. On će promatrane elemente slike podijeliti na tri skupine, tako što

će prvo postaviti granice b1 i b2. Ovisno o toj podjeli element slike biti će promatran kao oštećen ili neoštećen. Ako je klasificiran kao oštećen na njega će se primjeniti filtriranje. MATLAB kod posljednjeg algoritma podijeljen je na dvije (main i bdnd) funkcije, i njegovo izvođenje vremenski traje puno duže nego kod prethodno navedenih.

Operacijski sustav računala na kojemu se provode sljedeća testirana je 64-bitni Windows 7. Procesor AMD A4-3300M APU with Radeon HD graphics 1.90 Ghz.

Tab. 3.8. Vremena izvođenja filtriranja za Lena sliku s 20, 50 i 70% šuma

		Postotak šuma		
		20%	50%	70%
Vrijeme izvođenja	MF	00:00,3	00:00,5	00:00,6
	AMF	00:28,9	00:31,6	00:34,9
	BDND	12:36,0	13:47,8	14:57,0

Tab. 3.9. Vremena izvođenja filtriranja za Baboon sliku s 20, 50 i 70% šuma

		Postotak šuma		
		20%	50%	70%
Vrijeme izvođenja	MF	00:00,7	00:00,9	00:00,9
	AMF	00:28,5	00:32,3	00:37,2
	BDND	14:30,6	14:38,4	14:43,4

Tab. 3.10. Vremena izvođenja filtriranja za Pears sliku s 20, 50 i 70% šuma

		Postotak šuma		
		20%	50%	70%
Vrijeme izvođenja	MF	00:00,5	00:00,8	00:00,9
	AMF	00:40,2	00:43,1	00:49,1
	BDND	14:37,4	14:45,3	14:48,9

Tab. 3.11. Vremena izvođenja filtriranja za Barbara sliku s 20, 50 i 70% šuma

		Postotak šuma		
		20%	50%	70%
Vrijeme izvođenja	MF	00:00,6	00:00,7	00:00,7
	AMF	00:28,4	00:32,6	00:35,3
	BDND	14:34,3	14:40,1	14:44,7

Kod sve četiri test slike, sa svim postotcima šuma, izvođenje medijan filtra bilo je ispod jedne sekunde. Izvođenje nešto složenijeg adaptivnog medijan filtra povećava se na čak 30-50 sekundi za zadane postotke šuma, što je zadovoljavajuće vrijeme kod uklanjanja šuma. Adaptivni medijan filtar u kombinaciji s BDND algoritmom pokazao je razočaravajuće rezultate jer su se vremena izvođenja kretala od 12 do 15 minuta.

Kako bi se usporedilo vrijeme izvođenja AMF primjenom MATLAB programskog paketa i C++ provedeno je filtriranje na slikama Lena, Baboon, Pears i Barbara koje su oštećene s 20%, 50% i 70% šuma. Rezultati su prikazani u tablicama 3.12. do 3.15.

Tab. 3.12. Vremena izvođenja filtriranja za Lena sliku s 20, 50 i 70% šuma

		Postotak šuma		
		20%	50%	70%
Vrijeme izvođenja	AMF(M ATLAB)	00:28,9	00:31,6	00:34,9
	AMF(C++)	00:33,01	00:35,6	00:37,2

Tab. 3.13. Vremena izvođenja filtriranja za Baboon sliku s 20, 50 i 70% šuma

		Postotak šuma		
		20%	50%	70%
Vrijeme izvođenja	AMF(M ATLAB)	00:28,5	00:32,3	00:37,2
	AMF(C++)	00:31,9	00:35,6	00:41,9

Tab. 3.14. Vremena izvođenja filtriranja za Pears sliku s 20, 50 i 70% šuma

		Postotak šuma		
		20%	50%	70%
Vrijeme izvođenja	AMF(M ATLAB)	00:40,2	00:43,1	00:49,1
	AMF(C++)	00:42,9	00:48,7	00:54,4

Tab. 3.15. Vremena izvođenja filtriranja za Barbara sliku s 20, 50 i 70% šuma

		Postotak šuma		
		20%	50%	70%
Vrijeme izvođenja	AMF(M ATLAB)	00:28,4	00:32,6	00:35,3
	AMF(C++)	00:33,9	00:37,6	00:38,9

Prikazani rezultati pokazuju kako je u svim slučajevima vrijeme izvođenja AMF u C++ duže za nekoliko sekundi.

4. ZAKLJUČAK

U današnjem svijetu svakodnevno raste potreba za obradom slike, što se posebno počelo isticati u ogromnom porastu mobilnih aplikacija za uređivanje slika. Slike su često degradirane šumom prilikom njihovog prijenosa, ali i samog nastanka. U ovom radu su radu u programskom paketu MATLAB i u C++ napravljeni programi za uklanjanje impulsnog sol i papar šuma iz slika primjenom medijan filtra, adaptivog medijan filtra i adaptivog medijan filtra s BDND algoritmom. Cilj rada bio je napraviti programe koji će obuhvatiti različite medijan filtre, te usporediti efikasnost uklanjanja šuma primjenom pojedinog filtra Za mjeru kvalitete rekonstruirane slike korišten je PSNR.Rezultati filtriranja šuma su pokazali da je standardni medijan filtar vrlo dobar kod uklanjanja šuma koji je prisutan u slici do 20% elemenata slike oštećenih šumom, nakon čega njegova učinkovitost brzo opada. Adaptivni medijan filtar je pokazao vrlo dobre rezultate kod jednostavnijih slika, koje imaju jasne rubove te ne sadrže previše sitnih detalja. Adaptivni medijan filtar u kombinaciji s BDND algoritmom pokazao je bolje rezultate od adaptivnog medijan filtra kod slika koje sadrže više detalja i nejasnih rubova. Međutim, izvođenje posljednjeg algoritma u MATLAB-u traje predugo, čak i do 15 minuta, tako da kod uklanjanja šuma koje bi zahtijevalo brzo izvođenje, ovaj algoritam jednostavno nije zadovoljavajući. Također adaptivni medijan filtar izveden je i u programskom jeziku C++, te je pokazao jako dobre rezultate, ali nije prikladan zbog dugotrajne konfiguracije sustava i dosta zbunjujućeg povezivanja s OpenCv bibliotekama, koje često nisu prilagođene baš svakoj verziji Visual Studia, što često donosi puno problema, jer isti onda ne mogu biti pokrenuti sa svakog osobnog računala. Napravljeno je i iterativno uklanjanje šuma i to kroz deset iteracija. Rezultati su pokazali kako nakon četvrte iteracije vrijednosti PSNR-a, odnosno kvaliteta rekonstrukcija slika, rastu.

LITERATURA

[1] Bo Jin and Nam-Ho Kim, „A Modified Adaptive Switching Median Filtar for Image Restoration," 12.2.2007.
[2] Gonzales and Woods, „Digital Image Processing, 3rd ed.", Chapter 5, Image Restoration and Reconstruction
[3] Vedran Furač, „Uklanjanje šuma iz slike korištenjem dvodimenzionalne metode relativnog presjecišta intervala pouzdanosti," 5.9.2009
[4] Yiqiu Dong and Shufang Xu, „A New Directionl Weighted Median Filtar for Removal of Random-Valued Impulse Noise," ieee signal processing letters, vol.14,no.3, march 2007
[5] Sin Hoong Teoh and Haidi Ibrahim, „Median Filtering Frameworks for Reducing Impulse Noise from Grayscale Digital Images: A Literature Survey," International Journal of Future Computer and Communication, Vol. 1, No. 4, December 2012
[6]Peng Lei, „Adaptive Median Filtaring", Machine Vision, „mrežno", https://www.scribd.com/document/70512999/Adaptive-Median-Filtering, zadnji pristup 29.9.2016
[7] 2-D median filtaring, MathWorks, „mrežno", http://www.mathworks.com/help/images/ref/medfilt2.html, zadnji pristup 5.4.2016
[8] Install OpenCV 2.4.7 using Visual Studio 2010 / 2012, Youtube, „mrežno", https://www.youtube.com/watch?v=XvSnfV1cNjk, zadnji pristup 29.7.2016
[9] Impulse, gaussian and salt and pepper noise with OpenCV
, Stackoverflow, „mrežno", http://stackoverflow.com/questions/14435632/impulse-gaussian-and-salt-and-pepper-noise-with-opencv, zadnji pristup 29.7.2016
[10] Pei-Eng Ng and Kai-Kuang Ma, Senior Member, IEEE, „Switching Median Filtar With Boundary Discriminative Noise Detection for Extremely Corrupted Images", ieee transactions on image processing, vol.15, no.6, june 2006
[11] Peak signal-to-noise ratio, Wikipedia, „mrežno", https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio, zadnji pristup 1.9.2016

SAŽETAK

Standardni medijan filter poznat je kao učinkovito riješenje problema uklanjanja šumova iz slika, međutim kada se radi o slikama koje su teže oštećene šumma, standardni medijan filter gubi na učinkovitosti. U ovom je radu napravljen program za uklanjanje impulsnog sol i papar šuma primjenom medijan filtra, adaptivog medijan filtra i adaptivog medijan filtra s BDND algoritmom. Analizirana je kvaliteta slika nakon uklanjanja impulsnog šuma navedenim algoritmima te je utvrđeno da adaptivni medijan filtar učinkovito otklanja šum u slikama s manje detalja i s izraženim rubovima. Za slike s mnoštvom detalja najbolje rezultate pokazuje adaptivni medijan filtar s BDND algoritmom, ali ovaj način otklanjanja šuma zahtjeva značajno više procesorskog vremena i nije pogodan kod visokih postotaka šumom oštećenih elemenata slike.

Ključne riječi- Impulsni šum, adaptivni medijan filter, standardni medijan filter, detekcija šuma

ABSTRACT

Standard median filter is known as an effective solution in noise removing from corrupted images, however when it comes to badly corrupted images standard median filter loses its effectiveness. In this paper a program has been made to remove salt and pepper noise by applying a median filter, adaptive median filter and adaptive median filter with BDND algorithm. Quality of the image has been analyzed after removing the impulse noise using mentioned algorithms and found that adaptive median filter effectively eliminates noise in images with less details and with pronounced edges. For pictures with lot of details best results shows adaptive median filter with BDND algorithm, but this method of eliminating noise requires significantly more processing time and is not suitable for the large percentage of damaged picture elements.

Key words- impulse noise, adaptive median filter, standard median filter, noise detection

ŽIVOTOPIS

Adriana Matijas rođena je 26. listopada 1990. godine u Osijeku. Pohađala je osnovnu školu Svete Ane u Osijeku, nakon čega upisuje I.Gimnaziju u Osijeku, koju završava s vrlo dobrim uspjehom. Sudjeluje na natjecanjima iz područja fizike i informatike gdje pokazuje zanimanje za znanstveno-istraživačko područje rada. Nakon završene gimnazije upisuje izvanredni preddiplomski studiji računarstva na Elektrotehničkom fakultetu u Osijeku, gdje se posebno posvećuje proučavaju C programskog jezika i HTML. Na posljednjoj godini preddiplomskog studija počinje se baviti programiranjem mikrokontrolera, pri čemu je najveći praktični uspjeh postigla u izradi makete pokretne trake za odvajanje otpada temeljene na mikroupravljaču. Godine 2014. upisuje diplomski studij računarstva na fakultetu Elektrotehnike, računarstva i informacijskih tehnologija. Izradila je nekoliko web stranica, od kojih se najviše ističe web stranica praćenja ugroženih životinjskih vrsta. Trenutno se bavi proučavanjem Java Scripta i PHP-a, te radi na razvijanju mobilne aplikacije za prijedlog najpovoljnije kupovne košarice od ponuđenih i odabranih trgovina.

PRILOZI

Medijan filtar (MATLAB):

```
clc
clear all
close all
warning off

[FileName,PathName] = uigetfile('*.png','Select the Fundus image to test');
fullpathname = strcat(PathName,FileName);
Iin = imread(fullpathname);

Noise = input('Enter the percentage of salt&pepper noise : \n ');
Noise = Noise/100;
[m,n,o] = size(Iin);
if o == 3
 I = (rgb2gray(Iin));
else
 I = (Iin);
end

J = imnoise(I,'salt & pepper',Noise);
L = medfilt2(J,[3,3]);
if (size(I) ~= size(L))
 error('The size of the 2 matrix are unequal')

 psnr_Value = NaN;
 return;
elseif (I == L)
 disp('Images are identical: PSNR has infinite value')

 psnr_Value = Inf;
 return;
else

 maxValue = double(max(I(:)));
 mseImage = (double(I) - double(L)) .^ 2;
 [rows columns] = size(I);

 mse = sum(mseImage(:)) / (rows * columns)
 psnr_Value = 10 * log10( 256^2 / mse)
 RMSE = sqrt(mse)

End
```

Adaptivni medijan filtar(MATLAB):

```
clc
clear all
close all
warning off

[FileName,PathName] = uigetfile('*.png','Select the Fundus image to test');
fullpathname = strcat(PathName,FileName);
Iin = imread(fullpathname);
```

```
Noise1 = input('Enter the percentage of salt&pepper noise : \n ');
Noise = Noise1/100;
[m,n,o] = size(Iin);
if o == 3
 I = (rgb2gray(Iin));
else
 I = (Iin);
end
I1 = imnoise(I,'salt & pepper',Noise);



I2 = (padarray(I1,[3,3]));
[m1,n1] = size(I2);
Z = zeros(m1,n1);
c1 = 0;
c2 = 0;
for i= 4: m1-3
 for j = 4:n1-3
 flag = 0;
 for k = 3:2:7
 s = (k-1)/2;
 temp = double(I2(i-s:i+s,j-s:j+s));
 Smed = median(temp(:));
 Smin = min(temp(:));
 Smax = max(temp(:));
 A1 = Smed-Smin;
 A2 = Smed-Smax;

 if A1 > 0 && A2 < 0
 flag = 1;
 ss = s;
 c1 = c1+1;
 break;
 end
 end

 if flag == 1
 s= ss;
 temp = I2(i-s:i+s,j-s:j+s);
 Smed = median(temp(:));
 Smin = min(temp(:));
 Smax = max(temp(:));
 Sij = I2(i,j);
 B1 = Sij-Smin;
 B2 = Sij-Smax;

 if B1 > 0 && B2 < 0
 Z(i,j) = Sij;
 else
 Z(i,j) = Smed;
 c2 = c2+1;
 end
 end
```

```
 end
end
clc;
Z = Z(4:end-3,4:end-3);
squaredError = (double(I) - double(Z)) .^ 2;
MSE = sum(sum(squaredError)) / (size(Z,1) * size(Z,2));
PSNR = 10 * log10( 256^2 / MSE);
fprintf('\n The amount of noise added is %0.1f percent\n ', Noise1);
fprintf('\n The Peak-SNR value is %0.2f \n', PSNR);

subplot 131; imshow(uint8(I));title('Input image')
subplot 132; imshow(uint8(I2)); title('Input image with Noise')
subplot 133; imshow(uint8(Z)); title('Adaptive median filtared image')
```

Adaptivni medijan filtar u kombinaciji s BDND algoritmom, main funkcija (MATLAB):

```
clc
clear all
close all
warning off

[FileName,PathName] = uigetfile('*.png','Select the Fundus image to test');
fullpathname = strcat(PathName,FileName);
Iin = imread(fullpathname);
Noise1 = input('Enter the percentage of salt&pepper noise : \n ');
Noise = Noise1/100;
[m,n,o] = size(Iin);
if o == 3
 I = imresize(rgb2gray(Iin),[256 256]);
else
 I = imresize(Iin,[256 256]);
end
I1 = imnoise(I,'salt & pepper',Noise);


[MASK_BDND,Noise_Percent] = BDND_function(I1);
I2 = (padarray(I1,[3,3]));
[m1,n1] = size(I2);
Z = zeros(m1,n1);
c1 = 0;
c2 = 0;
for i= 4: m1-3
 for j = 4:n1-3
 flag = 0;
 for k = 3:2:7
 s = (k-1)/2;
 temp = double(I2(i-s:i+s,j-s:j+s));
 Smed = median(temp(:));
 Smin = min(temp(:));
 Smax = max(temp(:));
 A1 = Smed-Smin;
 A2 = Smed-Smax;

 if A1 > 0 && A2 < 0
```

```
flag = 1;
ss = s;
c1 = c1+1;
break;
end
end

if flag == 1
s= ss;
temp = I2(i-s:i+s,j-s:j+s);
Smed = median(temp(:));
Smin = min(temp(:));
Smax = max(temp(:));
Sij = I2(i,j);
B1 = Sij-Smin;
B2 = Sij-Smax;

if B1 > 0 && B2 < 0
Z(i,j) = Sij;
else
Z(i,j) = Smed;
c2 = c2+1;
end
end
end
end

clc;
Z = Z(4:end-3,4:end-3);
squaredError = (double(I) - double(Z)) .^ 2;
MSE = sum(sum(squaredError)) / (size(Z,1) * size(Z,2));
PSNR = 10 * log10( 256^2 / MSE);
fprintf('\n The amount of noise added is %0.1f percent\n ', Noise1);
fprintf('\n The Peak-SNR value is %0.2f \n', PSNR);

subplot 221; imshow(uint8(I));title('Input image')
subplot 222; imshow(uint8(I2)); title('Input image with Noise')
subplot 223; imshow(MASK_BDND); title('Noise Mask Detected using BDND')
subplot 224; imshow(uint8(Z)); title('Adaptive median filtared image')
```

Adaptivni medijan filtar u kombinaciji s BDND algoritmom, BDND funkcija (MATLAB):

```
function [MASK,Noise] = BDND_function(I1)

I2 = (padarray(I1,[10,10]));
[m1,n1] = size(I2);
Z = zeros(m1,n1);
OUT = I2;
MASK = zeros(m1,n1);
c1 = 0;
c2 = 0;
for i= 11: m1-10
 for j = 11:n1-10
 s = 10;
```

```
temp = double(I2(i-s:i+s,j-s:j+s));
Smed = median(temp(:));
SMED = Smed;
Sxy = I2(i,j);
Smin = min(temp(:));
Smax = max(temp(:));
temp = temp(:);
VO = sort(temp);
INDmed = find( VO == Smed);
VO = sort(temp);
for k = 1:(size(VO)-1)
VD(k,1) = VO(k+1) - VO(k);
end

templow = max(VD(1:INDmed));
temphigh = max(VD(INDmed+1 : end));
INDtemplow = find(VD == templow);
INDtemphigh = find(VD == temphigh);
IL = max(INDtemplow);
IH = min(INDtemphigh);
L = VO(1:IL);
M = VO(IL+1 : IH);
H = VO(IH+1:end);
clear VD
flag1 = 0;
flag2 = 0;
flag1 = sum(sum(ismember(L,Sxy)));
flag2 = sum(sum(ismember(H,Sxy)));
if flag1 == 0 && flag2 == 0;
MASK(i,j) =0;
else
s = 2;
temp = double(I2(i-s:i+s,j-s:j+s));
Smed = median(temp(:));
Sxy = I2(i,j);
Smin = min(temp(:));
Smax = max(temp(:));
temp = temp(:);
VO = sort(temp);
INDmed = find( VO == Smed);
VO = sort(temp);
for k = 1:(size(VO)-1)
VD(k,1) = VO(k+1) - VO(k);
end
INDmed = find( VO == Smed);

templow = max(VD(1:INDmed));
temphigh = max(VD(INDmed+1 : end));
INDtemplow = find(VD == templow);
INDtemphigh = find(VD == temphigh);
IL = max(INDtemplow);
IH = min(INDtemphigh);
L = VO(1:IL);
M = VO(IL+1 : IH);
H = VO(IH+1:end);
```

```
 flag1 = 0;
 flag2 = 0;
 flag1 = sum(sum(ismember(L,Sxy)));
 flag2 = sum(sum(ismember(H,Sxy)));
 if flag1 > 0 || flag2 > 0;
 MASK(i,j) =1;
 OUT(i,j) = Smed;
 end
 end
 end
end
MASK = MASK(11:266,11:266);
N = nnz(MASK);
Noise = (N/(256*256))*100
```

Adaptivni medijan filtar (C++):

```
#include "stdafx.h"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>
#include "opencv2/imgproc/imgproc.hpp"
#include <memory.h>
#include <vector>
using std::vector;

using namespace cv;
using namespace std;
int M,N,ch,m,n;

	double medianMat(cv::Mat Input, int nVals){

	float range[] = { 0, nVals };
	const float* histRange = { range };
	bool uniform = true; bool accumulate = false;
	cv::Mat hist;
	calcHist(&Input, 1, 0, cv::Mat(), hist, 1, &nVals, &histRange, uniform, accumulate);

	cv::Mat cdf;
	hist.copyTo(cdf);
	for (int i = 1; i <= nVals-1; i++){
		cdf.at<float>(i) += cdf.at<float>(i - 1);
	}
	cdf /= Input.total();

	double medianVal;
	for (int i = 0; i <= nVals-1; i++){
		if (cdf.at<float>(i) >= 0.5) { medianVal = i; break; }
	}
	return medianVal; }


int main( int argc, const char** argv )
{
 double xy;
	double min, max;
	cv::Mat img = cv::imread("C:\\Users\\Adriana\\Desktop\\AMF(new)\\abc\\abc\\lena.png",0);
```

```
        M = img.rows;
N = img.cols;
        Mat Inp = Mat::zeros(img.rows, img.cols,CV_8UC1);
        randu(Inp,0,255);
        Mat black = Inp < 40;
        Mat white = Inp > 210;

        Inp = img.clone();
        Inp.setTo(255,white);
        Inp.setTo(0,black);
        namedWindow("MyWindow", WINDOW_AUTOSIZE);
        imshow("MyWindow", img);

        namedWindow("MyWindow1", WINDOW_AUTOSIZE);
        imshow("MyWindow1", Inp);

        Mat Out = Inp;
        Mat temp1 = Mat::zeros(5, 5,CV_8UC1);
        Mat temp2 = Mat::zeros(7, 7,CV_8UC1);
        int c,r,i,j;

        for ( m = 3; m < img.rows-4; m++)
for ( n = 3; n < img.cols-4; n++)
{
                r = m-2;
                        for (i = 0; i <= 4; i++)
                        {
                                c = n-2;
                                for (j = 0; j <= 4 ; j++)
                                {
                                        temp1.at<uchar>(i, j) = Inp.at<uchar>(r, c);
                                        c++;
                                }
                                r++;
                        }


                        minMaxIdx(temp1, &min, &max);
                        double med = medianMat(temp1, 256);
                        double flag = 1;
                        if ((med == min) && (med == max))
                        {
                                r = m-3;
                                for (i = 0; i <= 6; i++)
                                {
                                        c = n-3;
                                        for (j = 0; j <= 6 ; j++)
                                        {
                                                temp2.at<uchar>(i, j) = Inp.at<uchar>(r, c);
                                                c++;
                                        }
                                        r++;
                                }
```

```
                    minMaxIdx(temp2, &min, &max);
                    double med = medianMat(temp2, 256);
                    double flag = 2;
                    if ((med == min) && (med == max))
                    {
                        continue;
                    }
                }

                if (flag == 1)
                {
                    xy = temp1.at<uchar>(2, 2);
                }
                if (flag == 2)
                {
                    xy = temp2.at<uchar>(3, 3);
                }

                if ((xy == min) || (xy == max))
                {
                    Out.at<uchar>(m, n) = med;

                }
                else
                {
                    Out.at<uchar>(m, n) = img.at<uchar>(m, n);
                }

            }


        namedWindow("MyWindow2", WINDOW_AUTOSIZE);
 imshow("MyWindow2", Out);


 waitKey(0);

 return 0;
}
```